Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

Tiago Filipe Lagarinhos Monte

AUTONOMOUS ELECTRONIC WATER METER BASED ON HYDROGENERATION

CONTADOR DE ÁGUA ELECTRÓNICO AUTÓNOMO BASEADO EM HIDROGERAÇÃO

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

Tiago Filipe Lagarinhos Monte

AUTONOMOUS ELECTRONIC WATER METER BASED ON HYDROGENERATION

CONTADOR DE ÁGUA ELECTRÓNICO AUTÓNOMO BASEADO EM HIDROGERAÇÃO

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre Engenharia Electrónica e Telecomunicações, realizada sob a orientação cientifica do Prof. Dr. Rui Manuel Escadas Ramos Martins, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais.

o júri

Presidente	José Carlos Esteves Duarte Pedro Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Orientador	Rui Manuel Escadas Ramos Martins Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Arguente	José Miguel Costa Dias Pereira Professor Coordenador do Departamento de Sistemas e Informática da Escola Superior de Tecnologia de Setúbal

agradecimentos

Gostaria de expressar a minha gratidão para com o meu orientador, Prof. Rui Manuel Escadas Ramos Martins, pela sua colaboração, ajuda, apoio e disponibilidade no desenrolar deste trabalho de mestrado.

Da mesma forma, gostaria de agradecer aos Engenheiros Carlos Alves e Cláudio Monteiro, e à restante equipa da empresas Globaltronic e HFA, pela forma acolhedora como me receberam e por todo o apoio que me prestaram sempre que necessitei do seu auxílio.

Por último quero agradecer à minha família, em especial aos meus pais e à minha namorada, pelo apoio que me têm dado ao longo de todo o meu percurso académico.

palavras-chave

Hidrogeração, microcontroladores, ultra condensadores, electrónica de baixa potência, telemetria

resumo

Esta tese tem como objectivo relatar o desenvolvimento de um sistema electrónico baseado em energias renováveis, mais especificamente, um contador de água electrónico baseado em hidrogeração.

O trabalho foi iniciado com um estudo das características do hidrogerador que iria ser utilizado, que foi desenvolvido pela Vulcano (sub-holding da Bosch) e utilizado em esquentadores de aquecimento de água para produzir a faísca de ignição.

De seguida, e após a selecção do método de armazenamento de energia (ultra condensadores), procedeu-se ao desenvolvimento de um sistema para demonstração da simplicidade na determinação do SOC nestes dispositivos. A carga é monitorizada por um microcontrolador e os dados são visualizados num computador pessoal através de um interface gráfico desenvolvido em C#.

Procedeu-se depois ao desenvolvimento do projecto de hardware e software do contador de água electrónico. Ambos foram feitos com o objectivo primário de obter o máximo de eficiência energética, mantendo uma boa performance geral do sistema. Fez-se ainda uma PCB para o hardware desenvolvido.

keywords

Hydrogeneration, microcontrollers, ultra capacitors, low power electronics, telemetry

abstract

This thesis aims to report the develolment of an electronic system based on renewable energies, specifically, an electronic water meter based on hydrigeneration.

The thesis work was initiated by studying the hydrogenerator's characteristics, which is a device developed by Vulcano (a Bosch sub-holding) and is used in water heaters to generate the ignition spark.

Following, and after selecting which energy storage method would be used, a charge monitoring system was developed. It aims to demonstrate how simple it can be to determine the SOC of ultracapacitors, when compared with batteries. The charging is monitorized by a microncontroller and the data is displayed on a computer, using a graphical interface developed in C#.

It was then developed the water meter's hardware and software project. Both had the primary objective of obtaining maximum energy efficiency, while maintaining a good overall system performance. A PCB was also developed for the hardware.

Contents

List of Figures

List of Tables

Nomenclature

AC Alternate Current

ADC Analog to Digital Converter

EDLC Electric Double-Layer Capacitors

GSM Global System for Mobile Communications

GUI Graphical user interface

HDG Hydro Generator

IDE Integrated Development Environment

LCD Liquid Crystal Display

LDO Low-Dropout

MPPT Maximum Power Point Transfer

NiCd Nickel-cadmium

NiMh Nickel-metal hydride

OC Open Circuit

PLC Power Line Communications

QFN Quad Flat No leads

ROC Rate of charge

RTC Real Time Clock

SC Supercapacitors

SOC State of Charge

UC Ultracapacitors

USART Universal Synchronous Asynchronous Receiver Transmitter

WF Water Flow

Chapter 1

INTRODUCTION

1.1 Motivation

From my own experience, the monthly water bill is usually an estimative, most of the times above the real value. To avoid overcharging, it is necessary a visual checking of the water meter by either the user or an employee from the service provider. There is always the chance that the user doesn't send the correct data, or simply doesn't send any data. Therefor, periodical readings must be done by an employee from the service provider, which comes, obviously, with some costs.

This problem could easily be solved using telemetry solutions. Furthermore, this solutions could consist of water meters able to transmit data remotely, eliminating the need of human intervention.

There are such water meters, but the drawback is that they depend on an external energy source (usually a battery pack). Batteries have a limited lifetime, so they required periodical replacement.

The motivation for this project is the development of an energetically self-sustained water meter. Such a device would have numerous advantages, namely, low maintenance needs, remote data transmission capabilities and being environmentally friendly.

1.2 Objectives

This thesis aims for the development of an autonomous electronic water meter capable of remote data communication, in order to eliminate human intervention in the reading process. The water meter will be based on a hydrogenerator developed by Vulcano (Bosch Subholder) and plays a double role in the system: energy source and WF measurement instrument. Its output is simultaneously used to measure the water flow (which will be explained further ahead) and to convert some of the water's kinetic energy in electrical energy.

An analysis of the hydrogenerator is required, to be conscious of its characteristics and capabilites, and taking the most advantage of this device to achieve maximum performance.

To make the water meter energetically self-sufficient, the converted energy must be stored either in batteries or other possible technologies, namely, UC technology.

Since it is a purely electronic system, it will have to be controlled by a microcontroller unit. The most recent generations of 8-bit microcontrollers have very low power consumptions, making them perfect for this kind of applications.

For remote data transmission, GSM seems to be the most reliable and universal solution. For this reason, a GSM modem will be included in the system. For local readings (if

necessary), the system will have a simple LCD display triggered by a reading button.

The final prototype is intended for the normal household of 3+ persons, being able to record all water consumptions and reporting them to the respective service provider. If intended, the user can make a visual checking of the water usage.

1.3 Thesis Structure

On this chapter the main objectives of this thesis are presented.

Chapter 2 is used to describe the state of the art regarding water meters and electrical storage technologies. It gives a general view on various types of commercially available water meters, divides them in categories and explains how the most common types work. Also, it describes the some of the characteristics of batteries and UC , followed by a discussion on which technology should be used for this project. Finally, the analysis of the hydrogenerator is presented, which includes field and laboratory testings and results.

On chapter 3 it is described a charge monitoring system for UC. This was developed with the goal of getting familliared with the UC technology and demonstrating how the SOC determination is done.

Chapter 4 describes the hardware development for the water meter prototype. Explains the main blocks and their functions.

Chapter 5 describes the software implemented, the method used for water metering, some of the more important functions and the general fluxogram of the system.

Chapter 2

STATE OF THE ART AND GENERAL CONCEPTS

2.1 Water Meters

Water Meters are devices used to measure the volume of water usage. In most developed countries, they are used at each residential and commercial building in a public water supply system. The total usage is usually displayed in cubic feet (ft^3), cubic meters (m^3) or US gallons. It can either be a mechanical or electronic register, and some water meters can also display instant water flow along with the total value.

They're used all over the world, and have become an important utility for some of the following reasons[1]:

- They make it possible to charge costumers in proportion to the amount of water they use;
- They allow the system to demonstrate accountability;
- They are fair for all costumers because they record specific usage;
- They encorage costumers to conserve water (especially as compared to flat rates);
- They allow a utility system to monitor the volume of finished water it puts out;
- They aid in the detection of leaks and waterline breaks in the distribution system.

Water meters are generally owned, read, and maintained by a public water provider such as a city, rural water association, or private water company.

Those used in typical water distribution system as designed for cold potable water only. For other types of water, special meters are used. For instance, hot water meters are designed with special materials that can withstand higher temperatures[2].

Meters are classified into two basic types, positive displacement and velocity, but with many variatons of each. There are also meters who combine both methods, called compound meters. These use a valve machanism to direct water flow into each part of the meter so readings can be taken from both mechanisms[3]. Follows a resume of some of the most used types of water metering systems.

2.1.1 Positive Displacement

Positive displacement meters are generally very accurate measuring low to moderate flow rates, therefore used in most homes, apartments, hotels and office buildings[3]. They operate by measuring water flow against a known volume of liquid held in a small chamber. The compartment moves with the flow of water and is repeatedly filled and emptied. The flow rate is then calculated based on the number of times this chamber is filled and emptied.

They usually have a built-in strainer to protect the measuring element from rocks or other debris that could stop or break the measuring element. Normally have bodies made of bronze, brass or plastic, with internal measuring chambers made from molded plastics and stainless steel[2].

The data is recorded using an arrangement of gears driven by a magnet, which is moved by either a nutating disc or an oscillating piston. This magnetic drive principle is used so that liquid doesn't come in contact with parts.

2.1.1.1 Nutating disc

Nutating disc (or wobble) meters are the most common type of displacement meters. This type of flow meter is normally used for water service, such as raw water supply. The movable element is a circular disc which is attached to a central ball. A shaft is fastened to the ball and held in an inclined position by a cam or roller. The disk is mounted in a chamber which has spherical side walls and conical top and bottom surfaces. The fluid enters an opening in the spherical wall on one side of the partition and leaves through the other side. As the fluid flows through the chamber, the disc wobbles (hence the name wobble) or executes a nutating motion. Since the volume of fluid required to make the disc complete one revolution is known, the total flow through a nutating disc can be calculated by multiplying the number of disc totations by the known volume of fluid.

The rotating motion of the disc operates a revolution counter, through a crank and a set of gears which is calibrated to indicate total system flow [1][4].

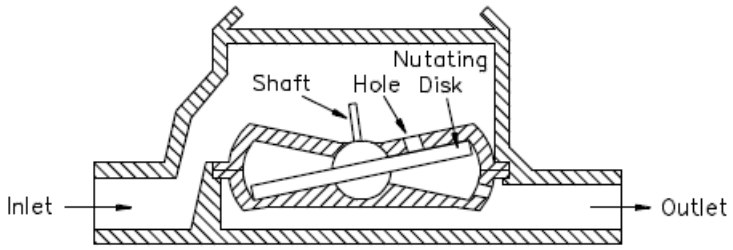


Figure 2.1: Nutating disc (taken from www.engineersedge.com)

2.1.1.2 Oscillating piston

Piston meters have a piston that oscillates back and forth as water flows through a precision-machined chamber containing that said piston. The position of the piston divides the chamber into compartments containing an exact volume. Liquid pressure drives the piston to oscillate and rotate on its center hub. The movements of the hub are sensed through the meter wall by a follower magnet. Each revolution of the piston hub is equivalent to a fixed volume of fluid, which is transmited to a register through an arrangement of magnetic drive and gear assembly[5].

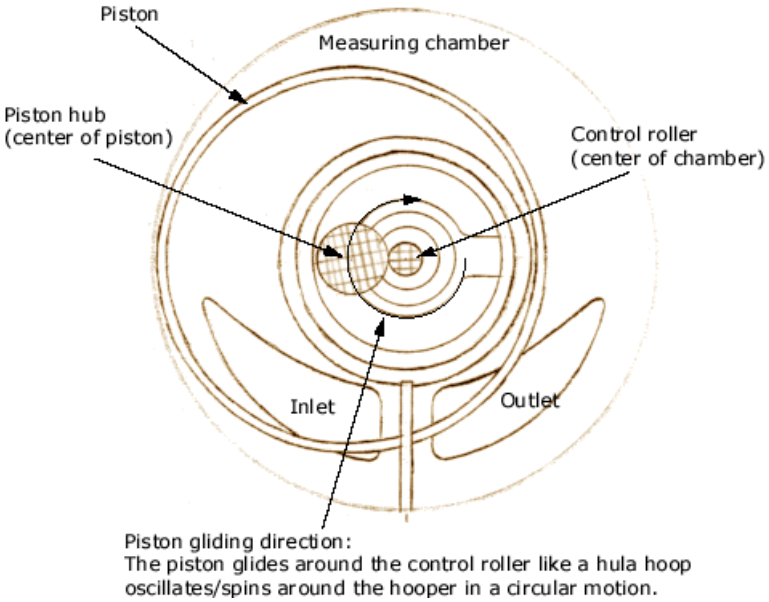


Figure 2.2: Oscillating piston (taken from www.smartmeasurement.com)

2.1.2 Velocity

Velocity meters operate on the principle that water passing through a know cross-section with a measured velocity can be equated into a volume of flow. They are mostly used for high flows of water, being used typically in large industries with a high water consumption. There are several types of velocity based meters: turbine, venturi, orifice, ultrasonic and magnetic, just do name a few. The ultrasonic and magnetic meters do not have mechanical measuring elements, they need electronic devices to make the measures[1, 2].

The following subsubsections contain a brief explanation of the refered water meter types

2.1.2.1 Turbine meter

Turbine meters have a rotating element that turns with the flow of water. Volume of water is measured by the number of revolutions of the rotor. The rotational speed is a direct function of flow rate and can be sensed by a magnetic pick-up, photoelectric cell, or gears[6]. They are less accurate at low flow rates.

The flow direction is generally straight through the meter, allowing for higher flow rates and less pressure loss than displacement type meters. They're used for large commercial users, fire protection, and as master meters for the water distribution system. Strainers are usually installed in front of the meter to protect the measuring element from gravel or other debris[1, 2].

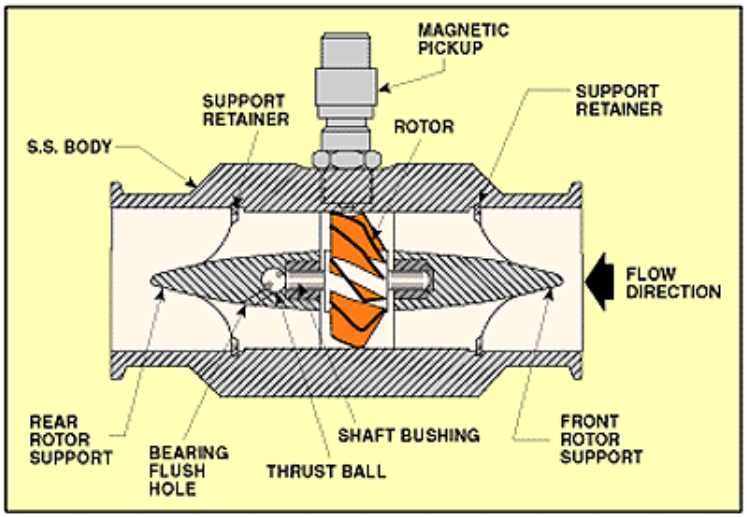


Figure 2.3: Turbine meter (taken from www.smartmeasurement.com)

2.1.2.2 Venturi meter

Venturi meters are used to measure the flow through a pipeline. They have a section that has a smaller diameter than the pipe on the upstream side. Based on a principle of hydraulics, as water flows through the pipe, its velocity is increased as it flows through a reduced cross-sectional area. Difference in pressure before water enters the smaller diameter section and at the smaller diameter “throat” is measured. The change in pressure is proportional to the square of velocity. Flow rate can be determined by measuring the difference in pressure. Venturi meters are suitable for large pipelines and do not require much maintenance[1].

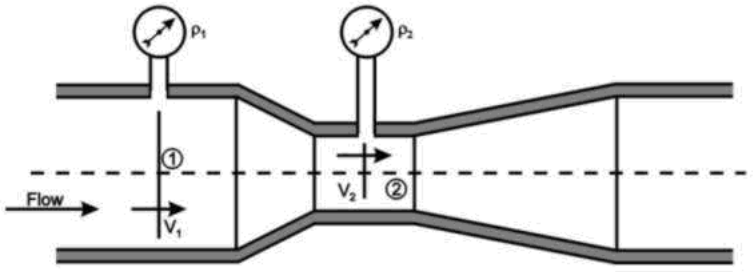


Figure 2.4: Venturi meter (taken from www.smartmeasurement.com)

2.1.2.3 Orifice meter

Like venturi meters, orifice meters are used to measure flow rate in a closed conduit (pipe). They also work on the same principle as venturi meters, except that, instead of the decreasing cross-sectional area, there is a circular disk with a concentric hole, an orifice plate. This orifice plate is thus installed into the pipe, sometimes using specialized “orifice plate changers” which conveniently hold the orifice plate in the pipe. Small access pressure ports, or pressure taps are required on each side of the orifice plate to allow the measurement of the pressure change across the plate when the fluid is flowing. Flow rate is calculated similarly to the venturimeter by measuring the difference in pressures[7].

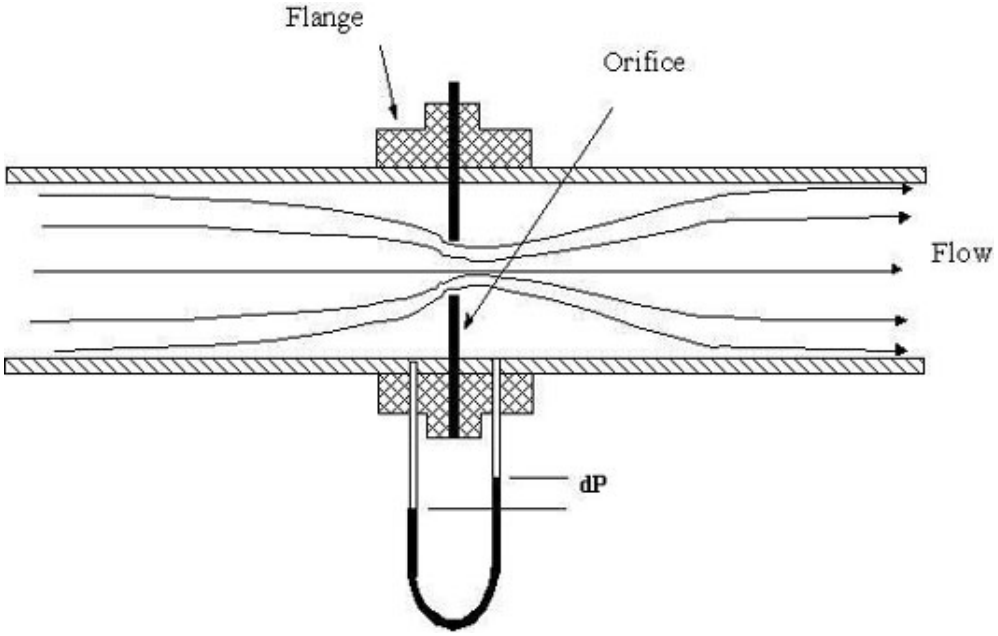


Figure 2.5: Orifice meter (taken from www.flowconditioner.com)

2.1.2.4 Ultrasonic meter

Ultrasonic meters send sound waves diagonally across the flow of water in the pipe. The basic principle of operation employes the frequency shift (doppler effect) of an ultrasonic signal when it is reflected by suspended particles or gas bubbles (discontinuities) in motion. Ultrasonic sound is transmitted into a pipe with flowing liquids, and the discontinuities reflect the ultrasonic wave with a slightly different frequency that is directly proportional to the rate of flow liquid. The difference in propagation delays of the ultrasonic waves is also a method used to measure the flow.

The results can be slightly affected by temperature, density or viscosity of the flowing medium. They are ideal for wastewater applications or any dirt liquid which is conductive or water based and will generally not work with distilled water or drinking water. Ultrasonic flowmeters are also ideal for applications where low pressure drop and chemical compatibility are required. They have no moving parts, therefore requiring very little maintenance[8].

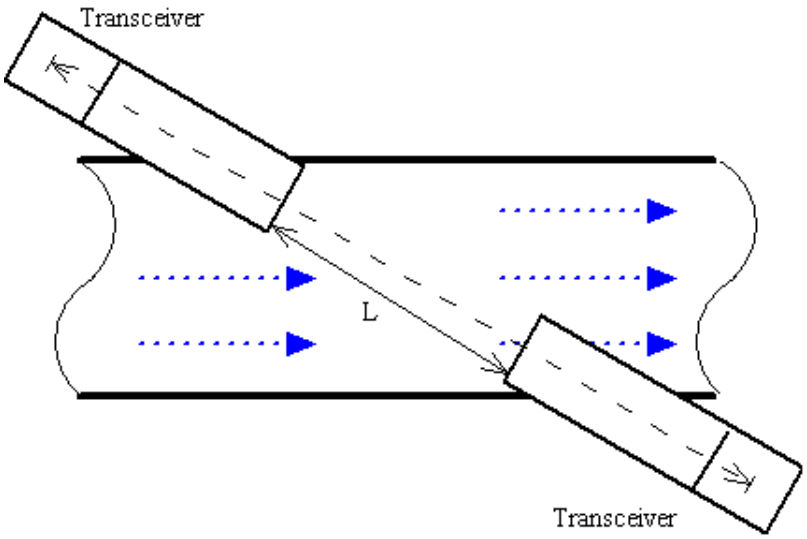


Figure 2.6: Ultrasonic doppler meter (taken from www.omega.com)

2.1.2.5 Magnetic meter

Commonly referred to as “mag meters”, they measure the water flow velocity by using electromagnetic properties. Specifically, Faraday’s law of induction is the basis for the measurement. A magnetic field is applied to the metering tube, which results in a potential difference proportional to the flow velocity perpendicular to the flux lines. Usually electrochemical and other effects at the electrodes make the potential difference drift up and down, making it hard to determine the fluid flow induced potential difference. To mitigate this, the magnetic field is constantly reversed, cancelling out the static potential difference. This however impedes the use of permanent magnets for magnetic flowmeters. It it also necessary to magnetically isolate the meter, so that the magnets don’t get affected by surrounding sources.

They have the advantage of being able to measure flow in both directions, and use electronics for totalizing the flow.

These meters can be used to measure “untreated” water, since there are no mechanical elements that might get damaged from debris[2][9].

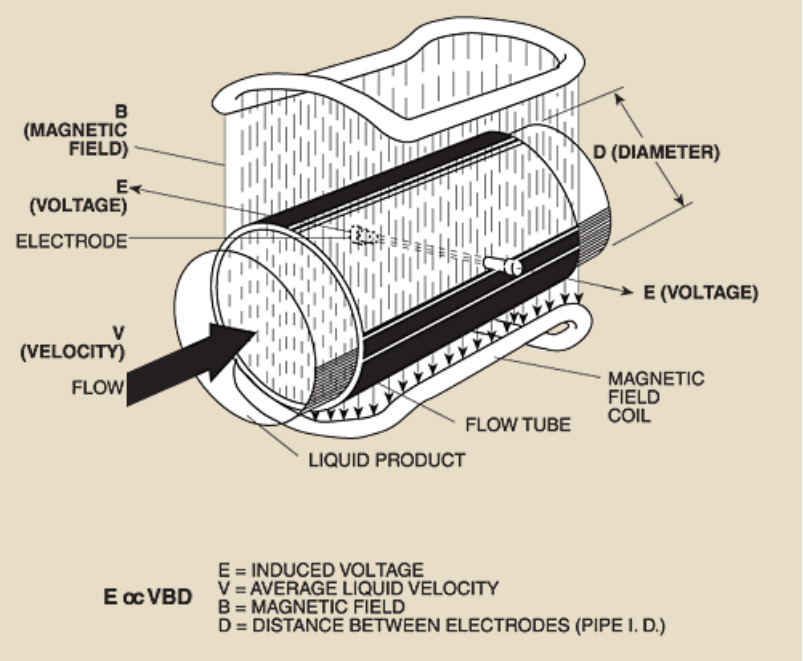


Figure 2.7: Magnetic meter (taken from www.wikipedia.org)

2.1.2.6 Electronic meter

While the two last meter types presented (ultrasonic and magnetic meter) are neither exclusively nor exhaustively electronic in nature, they do represent a logical grouping of flow measurement technologies, electronic meters. All have no moving partes, are relatively non-intrusive, and are made possible by today's sophisticated electronics technologies.

One the one hand, magnetic flowmeters are the most directly electrical in nature, deriving their first principles of operation from Faraday's law. On the other hand, ultrasonic meters owe their successful application to today's methods of digital signal processing.

Electronic meters have a number of advantages, some of which were already discussed previously. But perhaps the biggest advantage is that, by having the data stored electronically, one can employ remote data transmission methods. Such methods can use existing physical infrastructures, e.g. PLC, or they can be wireless, e.g. GSM, zigbee or bluetooth. This allows for water service suppliers to install a network of meters where the reading and taxing processes can be done automatically.

2.1.3 Remote transmission meters

There are some water meters equiped with remote data transmission tecnhologies.

The AquaMaster™ Explorer from the ABB company is an example of an electronic commercial water meter with GSM technology to provide wireless data transmission. The biggest drawback of this specific equipment is the need of an external battery pack.[10]

Two cities from the United States of America are currently installing automated water meter systems: New York and Houston. In New York's case, it is intended to have data readings on exact use on a monthly basis, and to have that data available on the internet. This system allows the city to save money, by not having workers to make estimatives every three months, and making a proper charging for the water service.

The city of Houston has had an operating wireless system since 1998, but the technology's range was short - about 90 meters - and relied on trucks to drive by and pick up the signal in order to read the meters[11, 12].

2.2 Electrical Energy Storage Systems

Electrical energy is an energy form based on the generation of a voltage difference between two points, allowing electric current flow when a charge is placed between said points. It's one of the most used energy forms by modern man, especially due to it's ease of transportation. It's mainly generated from thermal, hydro, wind and nuclear energy.

There are many technologies for storing electrical energy, most of which require transformation, which leads to energy losses due to the conversion process.

Therefore, storage systems can be divided in two main categories, and than subdivided into their specific storage reservoir. The next figure shows the main storage technologies and how they're grouped together.

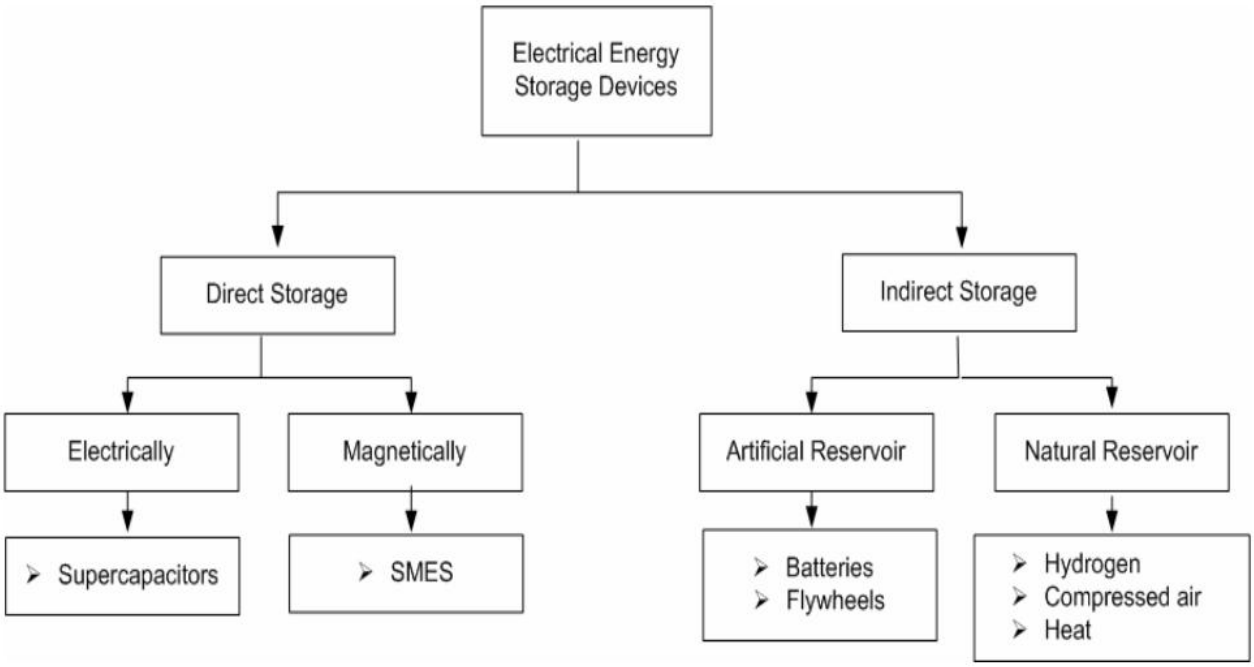


Figure 2.8: Electrical energy storage methods

Some of these technologies are intended for high power applications, some for low power, and others can be used on both. (i.e. batteries).

Modern man has witnessed the birth of the most diverse electronical equipment, some of which intended to be portable, and therefor "unplugged" from the power grid (PDA's, laptops, mp3 players,). A small, rechargeable and powerfull storage system is required, to provide the longest usage time between recharges.

The dominant energy storage technology for portable electronics are batteries. However batteries, like any other systems, have their cons. Their charging time is very long, have a short lifespan and are very sensitive to temperature. Other technologies are emerging and threatning to become a possible alternative to batteries. One of those emerging technologies are UC, which are already widely used as short term energy storage.

Both technologies will be presented in the next subsections, followed by a comparison and the conclusion over which technology to use for the project being developed.

2.2.1 Batteries

Batteries are electrochemical energy storage devices which convert chemical energy directly into electrical energy and vice-versa. They are formed by combining electrochemical cells of identical type in order to deliver higher voltage or current than a single cell. The battery cells create a voltage difference between the terminals of each cell, and hence, the battery's. When an external electrical circuit is connected to the battery's terminals, a current is driven[13].

Figure 2.9: Various batteries with different shapes and voltages (taken from www.wikipedia.org)

The modern battery was invented in 1800 by Alessandro Volta and has become a popular power source for many household and industrial applications. There are both rechargeable and non rechargeable batteries. For this application the interest lies, of course, on the rechargeable ones, since the system is meant to be autonomous. The recharging process consists of applying an electric current to the battery, which reverses the chemical reactions that occured during it's use.

All batteries are affected by self-discharge, meaning that even with no load applied, small chemical reactions occur, causing the battery to loose energy. The self-discharge rate is higher on rechargeable batteries than non-rechargeable.

Although rechargeable batteries may be refreshed by recharging, they still suffer degradation through usage. Normally a fast charge, rather than a slow overnight charge, will result in a shorter battery lifespan. However, if the overnight charger is not "smart" and cannot detect when the battery is fully charged, overcharging is likely, which will damage the battery[13]. By charging and discharging, the battery's internal resistance increases until reaching unusable values.

Since the introduction, the battery development effort has been concentrated on extending the capacity, increasing lifetime, minimizing self-discharge rate and increasing the power output while maintaining temperature stability, to mention some things[14].

The battery's characteristics depend greatly on what type of chemical composition it is designed with. The following table resumes some characteristics of the most used rechargeable battery types in terms of cell voltage, energy and power density, self-discharge rate and cycle durability[15, 16, 17, 18].

Chemical composition	Nominal cell voltage (V)	Energy density (Wh/Kg)	Power density (W/Kg)	Self-discharge rate (month^{-1})	Cycle durability
NiCd	1.24	40-60	150	10%	2000
Lead Acid	2.105	30-40	180	3%-20%	500-800
NiMh	1.2	30-80	250-1000	30%	500-1000
Lithium ion	3.6/3.7	100-160	250-340	5%-10%	1200

Table 2.1: Battery types and characteristics

Lithium ion batteries are the ones with higher enery density, which means that they can supply an electronic circuit for a longer period than any of the other types. They also have a longer lifespan, around 1200 charge-discharge cycles. This lifespan is strongly dependent on working conditions, specially temperature. On avergage, for each 10ºC above optimum working temperature, a battery losses 10% lifespan.

One other problem affecting battery's lifespan are incomplete charge/discharge cycles. The proper cycle should be taking the battery energy level from 90%/100% to 10%/20% during discharge and vice-versa during charge. Charging or discharging batteries with intermediate levels may decrease the total number of cycles before the battery becomes unusable. For the electronic water meter application in development, this is a huge drawback. Each time the HDG is activated (water starts flowing), the generated energy is conducted to the energy storage system, and when the HDG is inactive, energy is drawn from the storage to maintain the electronics. A full charge/discharge isn't likely to happen. For this reason, a battery would have an even shorter lifespan.

Another huge drawback of batteries is to determine its SOC. To estimate the energy level, a reading of the output voltage must be made, and the SOC can be determined using the known discharge curve (voltage vs. SOC). However, the voltage is more significantly affected by the battery's current (due to it's internal resistance) and temperature. Furthermore, a correction term must be calculated, proporcional to the battery current and temperature. Readings of current and temperature must be obtained, and compared with a table of the battery's Open Circuit Voltage Vs. Temperature. This adds up to some more electronics in the system, making it more expensive, complex, and less energetically efficient.

Since the HDG's output varies with the water flow, charge control must be implemented. As it shown later on, the HDG is capable of supplying over 100mA of current. Since most batteries should be charged with around 10% of their nominal current, it would be necessary to use a 1A battery to absorb the energy produced by the HDG. That would be an oversized battery for a system that is estimated to consume a few miliamps. If a smaller battery is used, most of the HDG's produced energy will be lost, with the risk of damaging the battery because of overcharging. This adds up to some more unnecessary complexity in the system.

2.2.2 Electric double-layer capacitors

Electric double-layer capacitors, also know as UC or SC, are electrochemical capacitors the have unsually high energy density when compared to common capacitors, typically on

the order of thousands of times greater than a high capacity electrolytic capacitor. As a comparison, a typical D-cell[19]sized electrolytic capacitor will have a capacitance in the range of tens of milifarads. The same size electric double-layer capacitor would have a capacitance of several farads, an improvement of about two or three orders of magnitude in capacitance, but usually at lower working voltage[20].

In a conventional capacitor, energy is stored by the removal of charge carriers, tipically electrons, from one metal plate and depositing them on another. This charge separation creates an electrical potential between the two plates, and a current will flow if connected to an external circuit.

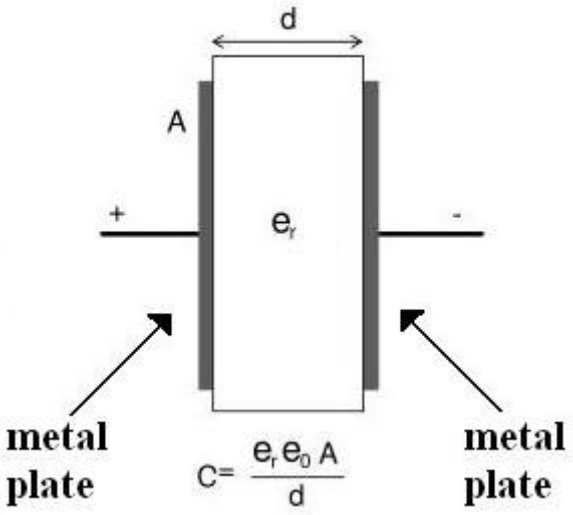


Figure 2.10: Capacitor diagram (taken from www.wikipedia.org)

The capacity is given by the formula $C = \frac{E_0 \times E_r \times A}{d}$ where:

- A - plate area [m^2];
- d - distance between plates [m];
- E_0- permittivity of free space = 8.854 x 10^{-12}[F/m];
- E_r- relative permittivity of the dielectric between the places [dimension less];

The capacity varies with the dielectric material between the plates, increases proportionally with the plate's area and decreasses with the distance between them. Optimizing the materials leads to higher energy densities for any given size of capacitor. However, the practical limitations on the surface area of the plates and the distance between the plates reduce capacitors' ability to have the same high levels of capacitance found in UC.

UC solve this problem by using nanoporous materials, tipically activated carbon, that allows them to have enormous surface area relative to the distance between the plates. When two pieces of activated carbon are immersed into a liquid electrolyte, they form an amazingly effective capacitor. The success of this method is primarily due to the carbon's large surface area-to-volume ratio, a product of the many microscopic nodules that cover its surface.

UC are made by coating two metal foil electrodes with this activated carbon, separating them with a thin piece of paper, and immersing the carbon-coated plates (the foil electrodes) into a liquid electrolyte.

The carbon on the negative plate collects electrons, which then attract positive ions from the electrolyte into the pores of the carbon. The carbon on the positive plate collects positive charges, which then attract negative ions from the electrolyte.

The thin piece of paper keeps the two plates from touching, which keeps the current from flowing between the two plates, allowing the positive and negative ions to move freely within their respective plates. This creates two layers of charge, making the ultracapacitor look like two capacitors in a series: this is the double-layer mechanism that gives ultracapacitors their name of "Electric Double-Layer Capacitors (EDLC)"

This design means that the charges on each plate of the ultracapacitor can be incredibly close to one another while still maintaining the carbon's large surface area, giving ultracapacitors their high capacitance[21].

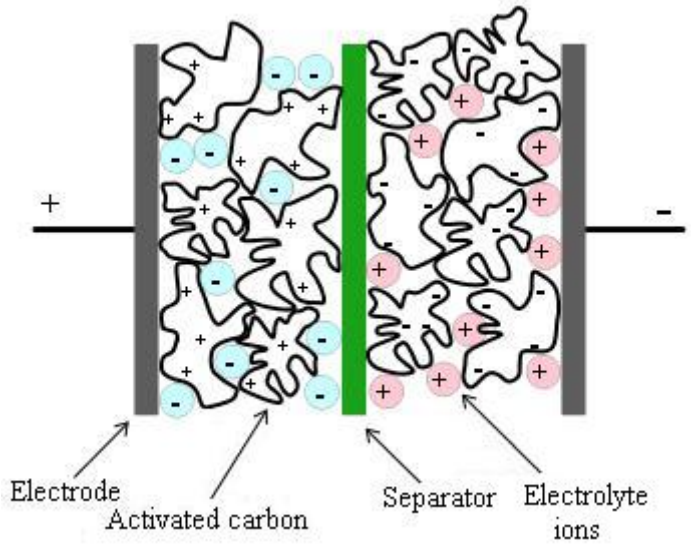


Figure 2.11: UC diagram (taken from www.wikipedia.org)

In terms of energy density, existing comercial UC range around 0.5 to 10 Wh/Kg. A group from MIT has demonstrated energy densities of 30 Wh/Kg and appear to be scalable to 60 Wh/Kg[22].

EEStor, a company from Texas, United States of America, claims to have developed a revolutionary new type of capacitor, with energy densities that could go over 250 Wh/Kg, intended to be used on electric cars. Many in industry have expressed great skepticism regarding these claims. As a matter of fact, no prototypes have yet been publicly independently tested or acknowledge by anyone outside the company.

In terms of power density, UC have 10 to 100 times higher power density than batteries. This characteristic makes them the number one choice in applications where high amounts of energy need to be rapidly absorbed/released. They can be either discharged or charged very quickly without being damaged, and can withstand more than 500,000 charge/discharge cycles. In terms of duration, UC usually last up to 12 years.

They also present a very low internal series resistance, very high conversion efficiency (up to 98%), extremely low heating levels and are less affected by temperature. There are no disposable parts during the whole operating life of the device, which makes it environmentally friendly.

The main drawbacks from UC are the linear discharge voltage, preventing the use of the full energy spectrum; low energy density when compared to state of the art batteries; low

cell voltage, meaning that serial connections are needed to obtain higher voltages and cell ballancing must be implemented to prevent over charging; higher self-discharged rates than batteries.

Regarding the self-discharge rate, it is said to vary from 0.1% to 3% a day. A self discharge test was performed by charging a 2.7 100F to approximatelly full capacity and measuring it's voltage during the course of 5 days. All the measurements were made at the same time of the day. The result is shown on table 2.2 .

Day	Voltage (V)	Energy (J)	Daily self-discharge (%)
1	2.6860	360.73	0%
2	2.6672	355.70	1.39%
3	2.6457	349.99	1.6%
4	2.6284	345.42	1.30%
5	2.6109	340.84	1.33%

Table 2.2: Self discharge test

The UC tested averaged 1.40% of self-discharge a day. The self-discharge was calculating based on the energy loss from one day to the following.

2.2.2.1 UC applications

The first ultracapacitor was developed by General Electric in 1957. The company used porous carbon electrodes to create the double-layer mechanism that characterizes ultracapacitors. The advantages of this mechanism weren't fully understood until nearly a decade later, when the Standard Oil Company of Cleveland patented their double-layer interface energy storage device in 1966. Ultracapacitors' first general use was as low-power, low-energy, long-life back-up power sources for consumer electronics such as VCRs. Today, UC are used with energy-storage and power-delivery functions in many different applications, which include:

- PC Cards
- flash photography devices in digital cameras
- portable media players
- energy recovery
- bridge power
- motor start up
- electric vehicles in general

UC are becoming very attractive for the automotive industry. China has developed an electric bus that runs without power lines, using power stored on a large onboard array of UCs.

This energy storage can only power the bus for an average of three miles, but it recharges very quickly in every bus stop by an elevating system placed on top of each bus. This "elevator" makes contact with two charging poles located above each bus station which supply the energy for the recharge. A backup battery system ensures 15 more miles, in case the UC loose all their power.

Two commercial bus routes are already using these buses.

Figure 2.12: The UC bus at a bus stop (taken from www.youtube.com)

Honda's hydrogen car also uses a large bank of ultra capacitors to supply short and high power bursts, during start up or acceleration. This hybrid battery/UC energy storage system uses the UC to buffer the batteries and protect them from high current demands, ensuring a longer lifespan.

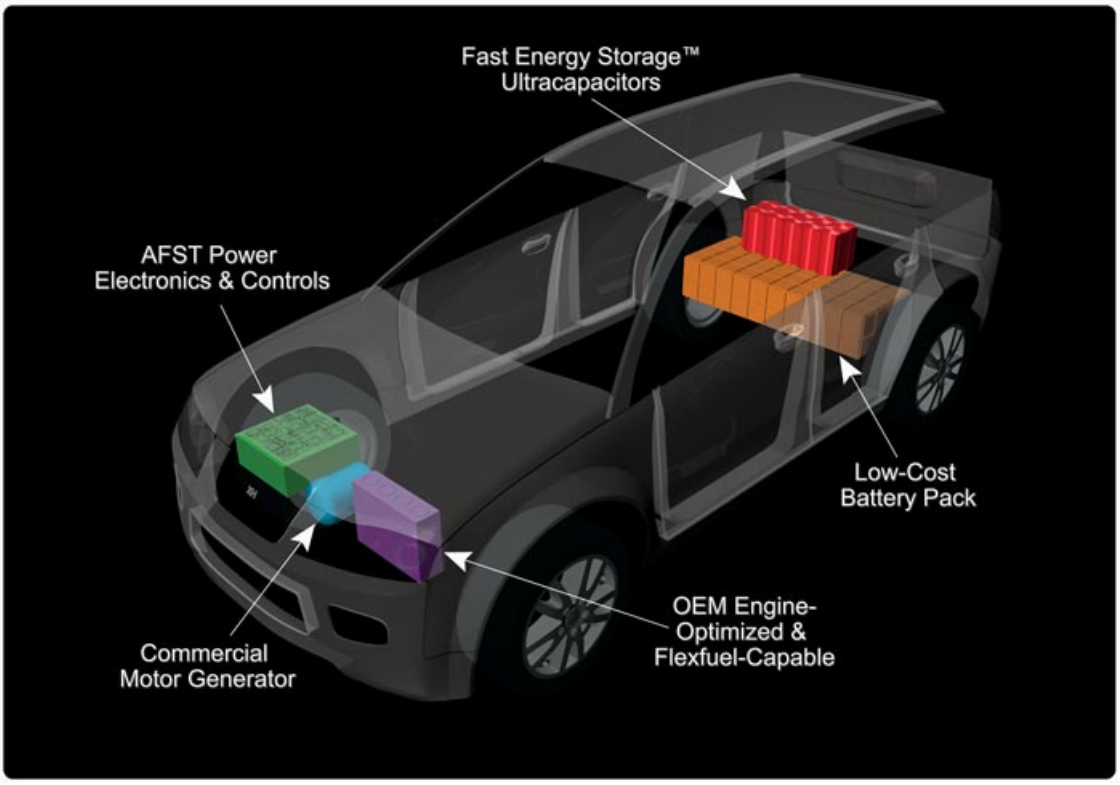


Figure 2.13: Batteries/UC hybrid energy storage system (taken from www.afstrinity.com)

In addition, the UC are used to capture the energy generated during breaking, which would otherwise be lost in overheating. This technique is called regenerative breaking.

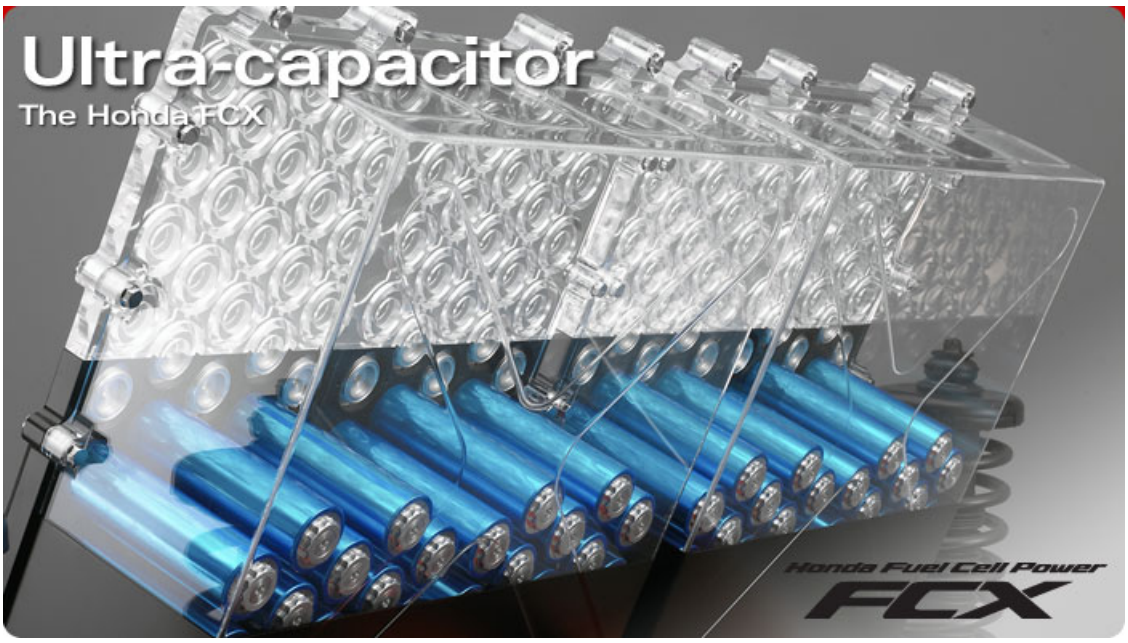


Figure 2.14: Honda's FCX UC bank (taken from www.honda.com)

2.2.3 Choosing the energy storage system

After studying the characteristics of both batteries and UC, let us proceed with a comparison between them, in order to choose the most suitable solution.

The two following tables summarize the main pros and cons of batteries (in general) and UC:

Pros	Cons
Widely used technology	Temperature sensitive
Relative high energy density	Limited and controlled ROC
Flexibility	Limited lifespan
Voltage is independente from SOC	Limited charge/discharge cycles
Relative low self-discharge	Difficult SOC estimation

Table 2.3: Batteries pros and cons

Pros	Cons
Over 500,000 charge/discharge cycles	Relative low energy density
Very high ROC	Voltage change with SOC
Easy SOC estimation	Sensitive to overcharging
High Power density	Relative high self-discharge
Low internal resistance	Low voltage

Table 2.4: UC pros and cons

The following graph resumes the difference between UC and batteries, when it comes to Energy and Power Density.

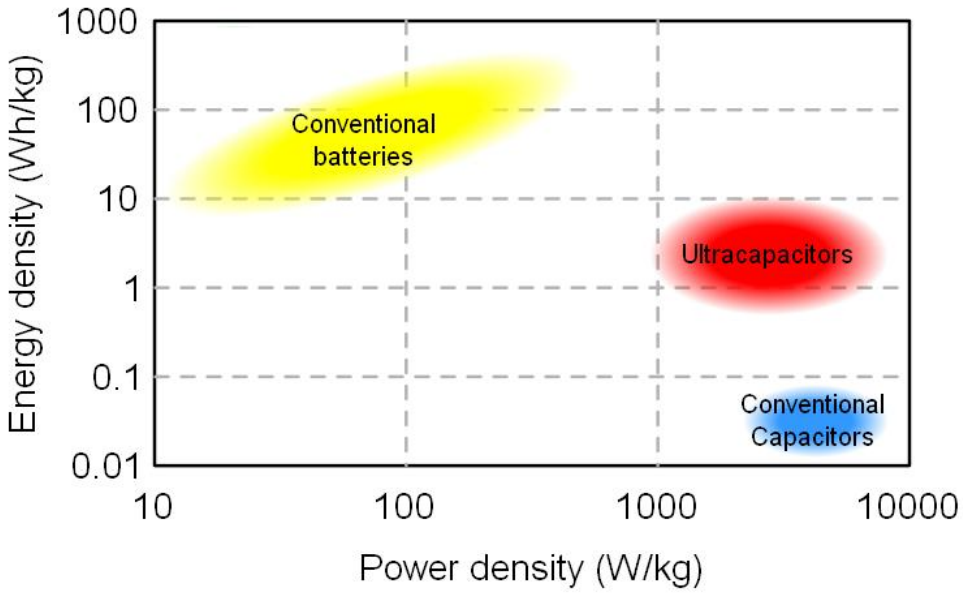


Figure 2.15: Energy and Power Density (taken from www.wikipedia.org)

The key features of the energy storage system to be implemented should be:

- Long lifespan (or number of cycles), in order to eliminate the need of system maintenance;
- The ability to charge the device with least control as possible, so it can be connected to the HDG with the minimum charge interface electronics and absorb all the energy produced;
- Easy SOC estimation, also to make the system simpler both in terms of hardware and software;

High energy density isn't a main characteristic, since the HDG will produce energy every time the water flows. Also, we don't need high voltages (above 5V), since the system will be very low power, and most components are supplied by voltages from 2V to 5V. Since there are different working voltages for different components, there will be the need for some voltage regulators, hence, having a voltage that changes with SOC isn't a problem.

The self-discharge should be the lowest possible. If the system looses its energy, there may be a period necessary for a minimum charge before the meter starts working.

2.2.3.1 Conclusion

From this short analysis, UC seem to be the right choice as an energy storage system.

- They can be charged at any rate without being damaged, hence, no need for charge control, which leads to system simplicity;
- SOC can be determined simply by measuring the output voltage of the UC and they are able to withstand over 500,00 charge/discharge cycles, which translates into years of use without the need of replacement;
- Number of charge/discharge cycles and overall lifespan will also increase the system's lifespan;
- Higher self-discharge of UC can be a problem, but if the meter works on a daily basis, a few minutes of running water may be enough to compensate for that energy loss.

2.3 Hydro generation

2.3.1 History of hydro generation

Throughout history, people have always used water to their advantage. Romans had the aqueducts, the Egyptians were masters in irrigation methods, and around 200 B.C. the first water wheel was built. It converts the kinetic energy of the moving water into mechanical energy and was initially used to mill flour in watermills, but also had some other purposes such as foundry and machining, and later, hydro electricity. The first use of moving water to produce electricity was a water wheel on the Fox River, Wisconsin, in 1882, which was shortly followed by a plant at the Niagara Falls[23]. It is estimated that hydroelectricity represents approximately 20% of the total worldwide electricity production and 88% when it comes to renewable sources[24]. Hydro generation can be divided into five main categories, according to their production capacity:

- Large Hydro: more than 10MW, usually supply energy for a few dozen small cities;
- Small Hydro: between 10MW and 1MW, can supply a city or some industrial plants;

- Mini Hydro: between 1MW and 100KW, can supply a small community or small enterprises;
- Micro Hydro: between 100KW and 5KW, can supply either a small community or a single household.
- Pico Hydro: less than 5KW, which can be enough to supply an entire household at least some of the home appliances.

Large Hydro Plants are usually located in big river streams, and the energy generated is then distributed to the surrounding area. As for all the other installations, tend to take advantage of smaller river streams, waterfalls, or any other source of running water. They also tend to be located as close as possible to the community/factory/house to be supplied, which guarantees a higher transport efficiency. Early hydroelectric power plants were much more reliable and effective than fossil fuel based power plants, which resulted in the proliferation of hydroelectric plants worldwide. In the middle of the 20th century, due to the increase of electricity demand and the incerasing efficiency of coal and oil fueled power plants, smaller hydroelectric plants fell out of favor. Nowadays, hydroelectric power plant projects are more focused in gigantic size plants, capable of producing more than 2000MW. The Three Gorges Dam in China is expected to reach 22500MW of electrical power production when it becomes fully operational in 2011. It's located on the Yangtze River, and its construction began in December 1994[25].

Figure 2.16: Three Gorges Dam (taken from www.wikipedia.org)

2.3.2 The world's smallest hydro generator

Vulcano is a sub holding company of the Bosch Group, specialized in the development and production of heating solutions, such as gas water heaters, central heating equipment and solar panels. In June 2000, Vulcano presented the world's smallest hydro generator. Its purpose was to power up the automatic ignition and supply the electronics, instead of the common 9V batteries. Every time the hot water was turned on, the hydro generator's rotation generated the energy required to the ignition process. This gave birth to a new gas water heater model, the Click HDG, which is energetically self-sustained, more ecologic, reliable and economic[26, 27]. The next figure shows the referedhydro generator.

Figure 2.17: Vulcano hydro generator (taken from www.vulcano.pt)

As the water passes through the hydro generator, there's a small bypass that takes a part of that water and uses it to spin a rotor, which is no more than a cylindrical magnet. The magnet's movement generates (by magnetic induction) a current in the stator (a circular winding) and a subsequent AC voltage in the output leads. The AC signal's frequency and amplitude are proportional to the water flow, and therefore can be used as a flow measuring instrument.

To use the HDG as a flow measurement instrument, a mathematical relation must be established between either the frequency or the amplitude of the signal and the flow of water. In order to do so, a study of the HDG's characteristics was performed.

It was connected to a hose so that the measures could be taken on real working conditions. Since there was no flowmeter available, the method used to measure the water flow was filling a half liter bottle and counting the filling time. From that value, and using a simple mathematical rule, the flow meter in liters per minute could be calculated.

On the next subsection the results of the study are presented,

2.3.3 Hydro generator study

2.3.3.1 AC characteristics

To have a mathematical model of the HDG, the first test consisted on measuring the output of the HDG (peak-to-peak voltage and frequency) using a digital oscilloscope. As previously mentioned, this test was performed in real conditions and with different running water flows. The output was left as an OC, with no load connected. The following graphs resulted from the measurements.

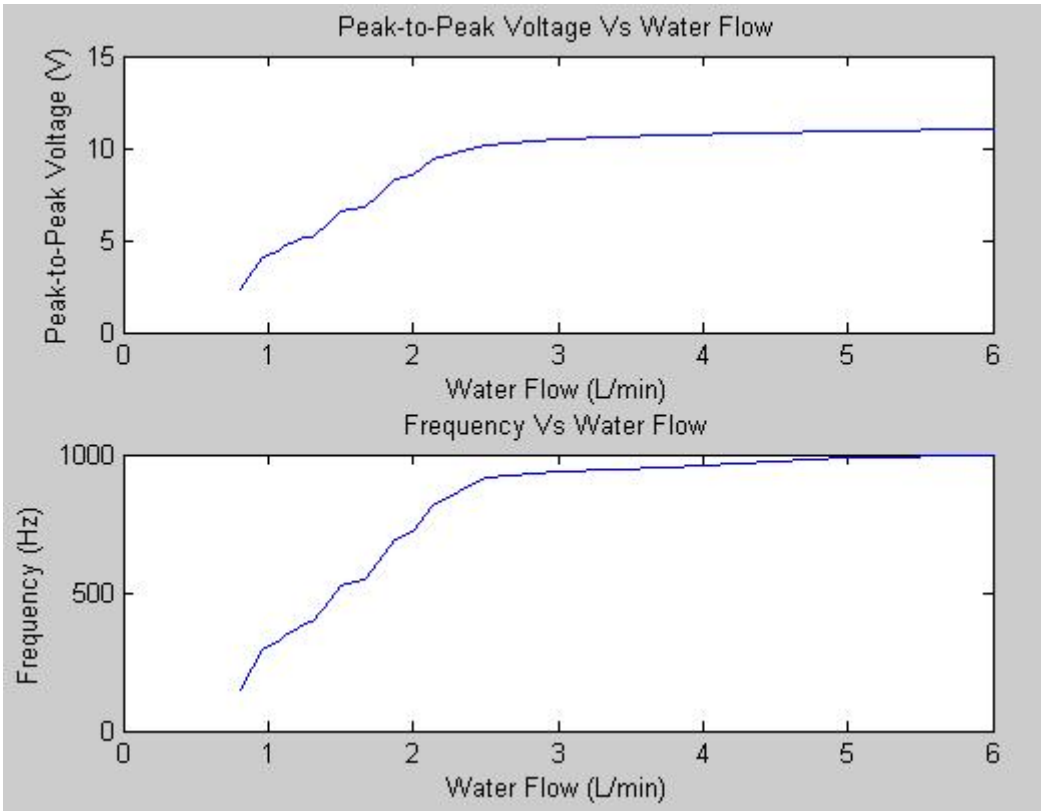


Figure 2.18: Frequency and peak-to-peak voltage Vs Water flow

It is clear the the HDG has a saturation point, where the increase of water flow doesn't produce an increase of the output signal, neither in voltage nor frequency. That saturation point will be set at 2.5L/min. Another conclusion is that te HDG needs a minimum of 0.8 L/min of water flow to start opreating. Between 0.8 and 2.5 L/min, both voltage and frequency respond linearly to the water flow increase. This is a typical response of an AC generator.

The saturation is due to the size of the water bypass described earlier, and not to the generator's physical properties, as it will be proved later on (with a DC motor simulating the water flow).

In order to find the mathematical relation between the water flow and the signals characteristics, the data was analyzed only on the linear response zone, ranging from 0.8 to 2.5 L/min. Using Matlab fitting tools, a linear fitting was performed on the data to obtain the mathematical equations that will be used.

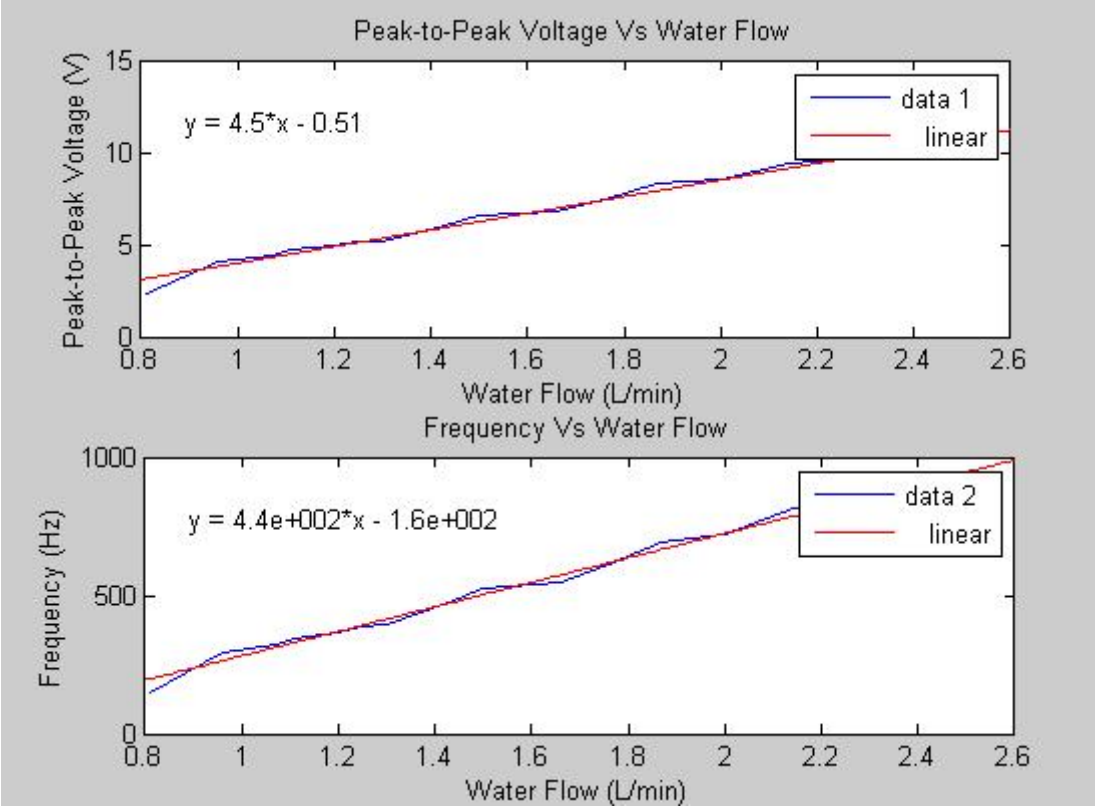


Figure 2.19: Frequency and peak voltage Vs Water flow (up to 2.5 L/min)

From the fitting process resulted the two following equations:

- $V = 4.5 \times WF - 0.51$, the relation between the WF and the output voltage (V in Volts and WF in L/min)
- $F = 4.4e^2 \times WF - 1.6e^2$, the relation between the WF and the output frequency (F in Hz and WF in L/min)

These equations can be used to calculate the WF based on the readings of frequency and/or voltage.

2.3.3.2 Laboratory testing

Since no closed water circuit is available, a solution had to be found to use the HDG in laboratory. The solution consisted in disassembling an HDG and place a small DC motor to spin the rotor magnet. In fact, that solution was already available since the HDG was used in previous projects by the company Globaltronic, where this project was developed.

When using this setup to simulate the water flow, it was noticeable that for a given frequency, the peak-to-peak voltage was smaller than on the HDG used with running water. This could be caused by the following reasons:

- the HDG had not running water, therefor no cooling of the coil. An increasing temperature can affect the output voltage by decreasing it;

- also, this HDG has been extensively used on previous testing, so it must have some degree of degradation.
- it is also that the motor doesn't exacly simulate the WF, an this causes a different output than the one obtained with water.

Also, the conclusion drawn from this observation is that the mathematical model that relates WF with output voltage cannot be used to in the reading method. The frequency Vs WF equation is the one to use in the water metering.

2.3.3.3 Power factor correction and maximum power transfer

In order to obtain the maximum output power, a power factor correction must be done. To do so, several capacitors were connected in parallel with the HDG's output. Also, and to find the maximum power transfer point, several resistors were also placed, but separately from the capacitors. All the measurements were done with the same simulated water flow, so the motor used for the simulation was always supplied with the same voltage. This test will also allow for an estimation of the internal impedance of the HDG. The next graphs show the peak-to-peak voltage and RMS voltage for both the capacitors and the resistors test. The reason to measure the RMS is that the capacitors change the shape of the output signal from the HDG (which is originally a perfect sine wave).

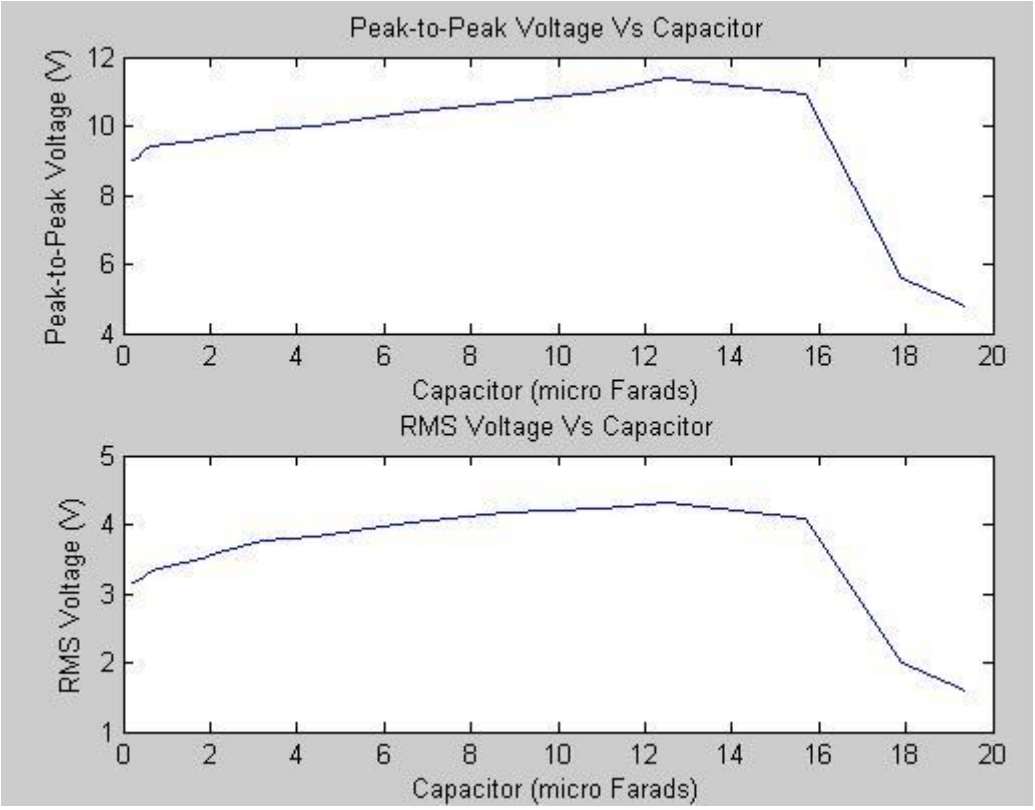


Figure 2.20: Output voltage with different capacitors

The capacitor that gives the bigger boost to the the output voltage is the 12.5uF capacitor. Therefore, to conduct the resistos test, this capacitor was also placed in parallel with the output, in order to draw more power. Several resistors were connected and only the RMS voltage was measured, which was then used to calculate the power transferred to the resistor ($P = V^2/R$).

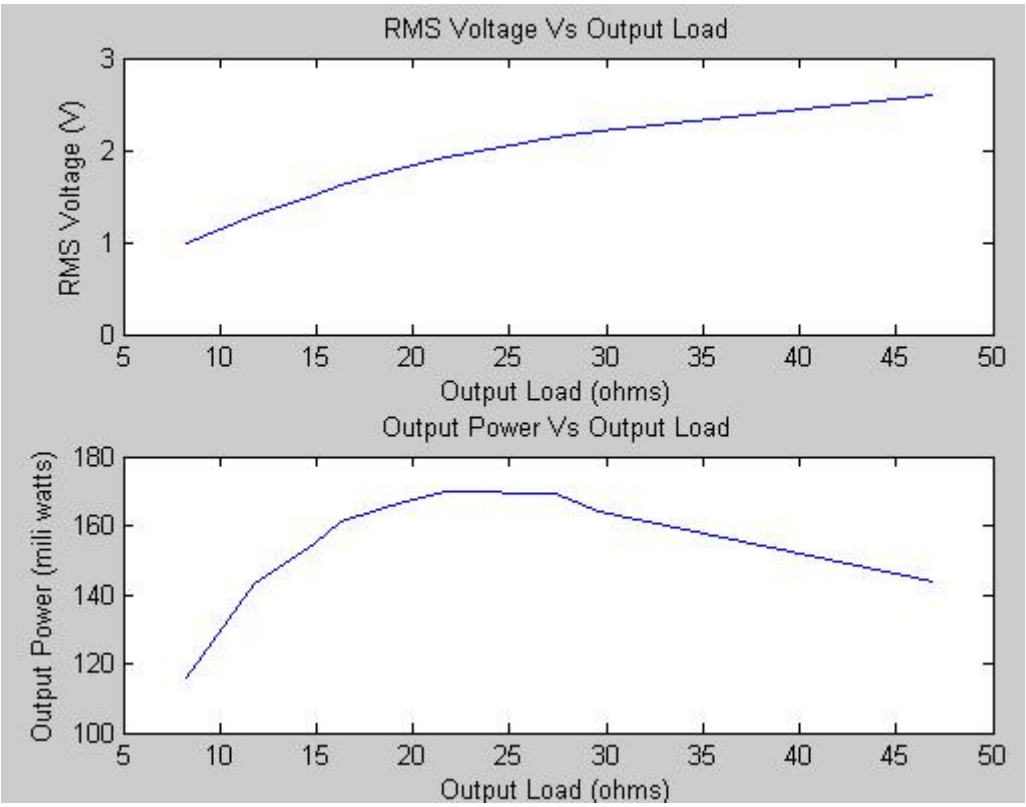


Figure 2.21: Power transfer

The maximum transferred power is when the output resistor is between 20 to 25 ohm, more specifically, 21.7 ohm with 170 mW of transferred power. With this data, an estimation of the internal impedance of the HDG can be calculated. This is a representation of the circuit:

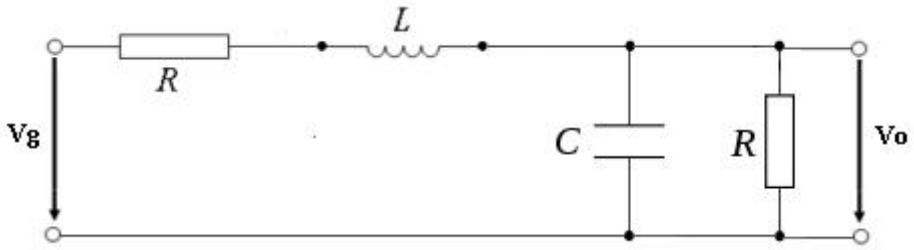


Figure 2.22: Equivalent circuit with parallel load impedance

The internal impedance of the generator (Zg) is the R and L series and the impedance of the load applied (Zo) is the R and C parallel. It is known that for a maximum power transfer, Zo must be the conjugate of Zg. Assuming that C=12.5uF and R=21.7ohm are the values corresponding to the maximum power transfer, the parallel model for the load must be converted into the series model:

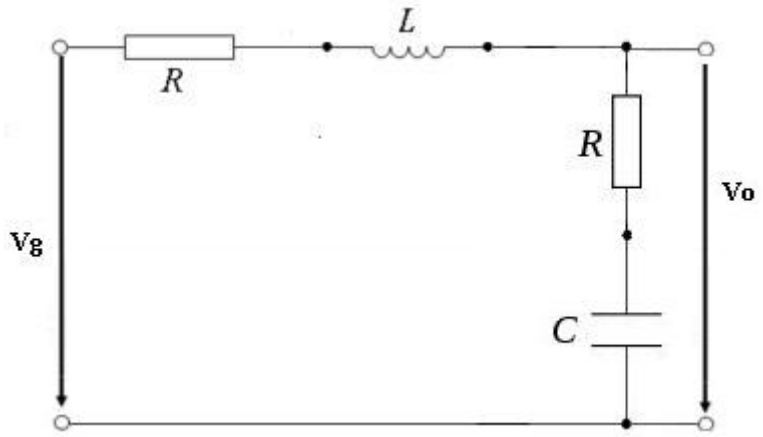


Figure 2.23: Equivalent circuit with series load impedance

After converting from the parallel to the series model, the new values for R and C are R=6.48 and C=17.8uF, thus the internal R=6.48 and internal L=1.7mH.

To charge an UC using the HDG, the signal must be rectified. To do so, a diode bridge was inserted, and several capacitors were connected parallel with the output of the HDG. No load was placed after the rectifier. It is assumed that HDG generates more energy with a correction capacitor, because of the voltage boost.

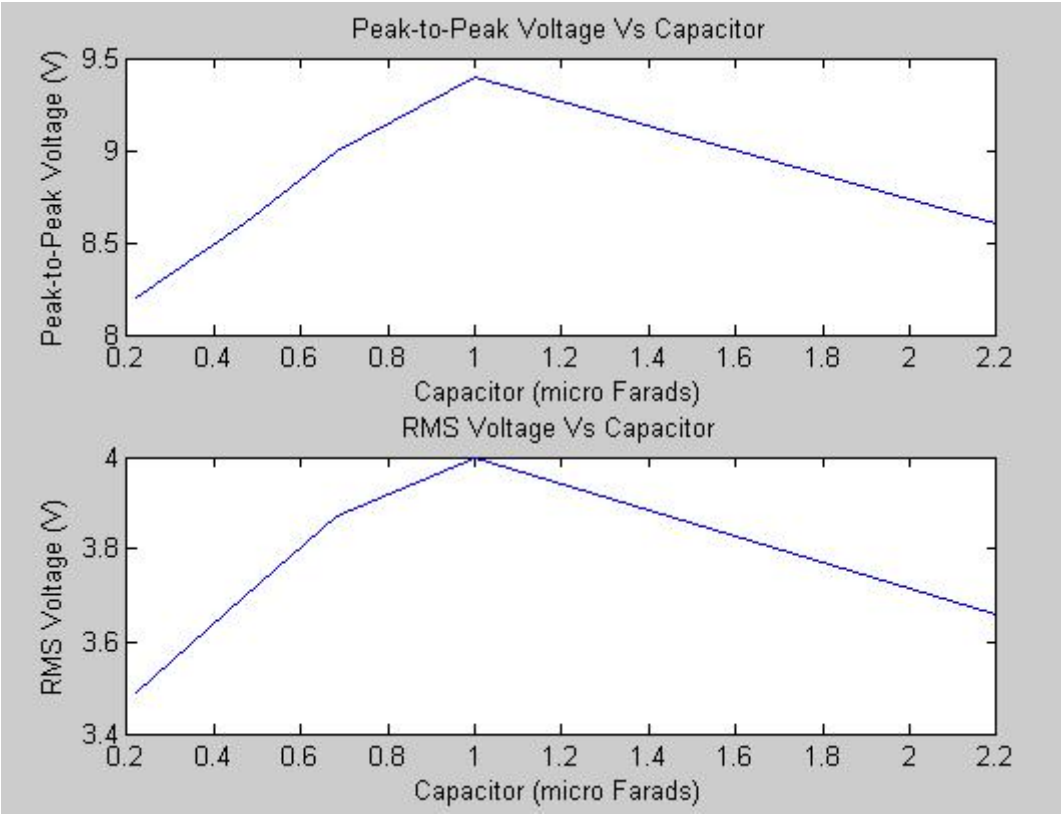


Figure 2.24: Output capacitors

Now it can be observed that the capacitor that gives the output voltage the bigger boost is the 1uF, and thus, a different capacitor from the previous test. The RMS voltage was measured at the output of the rectifier circuit with no load connected. At this point, the decision was made to use the 1uF capacitor as the power factor correction capacitor.

Since the purpose of the HDG is to charge an UC, the final test was charging a 100F 2.5V UC with the HDG. The ultra cap was connected directly to the output of the rectifier, so it could drain a high current in the initial charging stage. Measures of current and voltages (HDG's peak-to-peak voltage and UC voltage) were taken every 30 seconds. To measure the current a 1ohm resistor was placed in series with the UC. The HDG's rotation was kept constant during this test with a value of 800Hz.

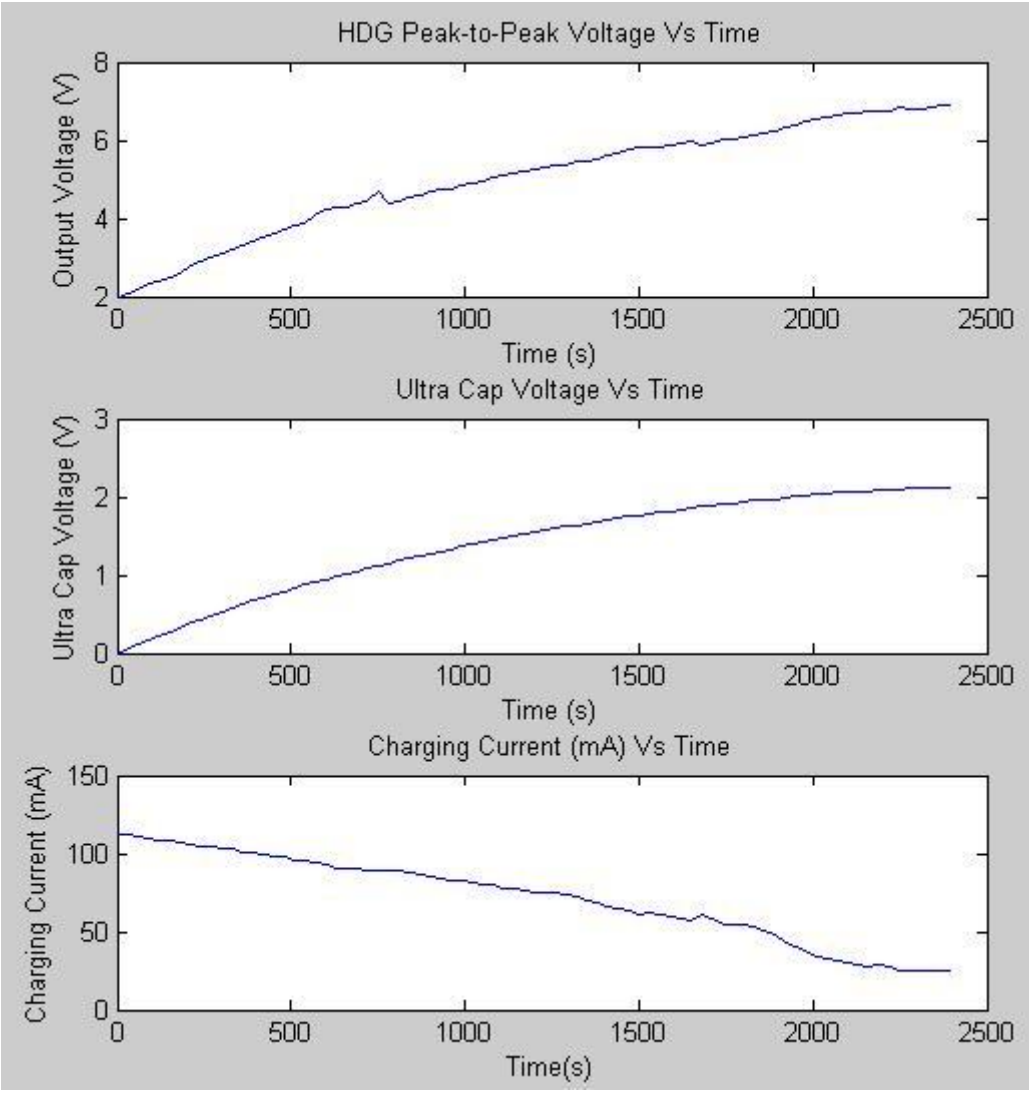


Figure 2.25: Ultra cap charging

As expected, the ultra cap reveals a logarithmic charging behavior, also because it is charged with a variable current. Both the output peak voltage of the HDG and the charging current vary throughout the charging test, which was also expected, since the HDG voltage is limited by the UC voltage. The initial current was 120mA an dropped to 30mA after 40min of continuous charging.

If a single hydro generator is used in a household, with family of four, and assuming that everyone takes a 10 min bath ever day, the produced electrical energy would be enough to charge one ultra cap up to 2V. Additional water usage (for laundry or dish washing) would even increase the charge on the ultra cap.

Chapter 3

UC CHARGING AND SOC MONITORING SYSTEM

3.1 Introduction

It is a well known fact that battery chargers can be quite complex, both in terms of hardware and software. They must have current, voltage and temperature control, in order to charge the batteries without causing damage. As a first approach to UC based electronics, a charging and monitoring system was developed. It is based on a Microchip microcontroller (PIC18F14K50) which reads and converts current and voltage values. Those values are then sent, to a computer, using an RS-232 interface. The computer later processes and presents the data in a graphical interface developed in C#. Charging current, UC voltage and energy level are shown in a friendly and easy to read manner. To make communication possible between the microcontroller and the computer, a device was used to convert between TTL/CMOS levels and RS-232 levels which is this case was an SP232ACP. For the results displayed, a 100F 2.5V UC was used

3.2 HDG and rectifying bridge

The HDG is an AC voltage generator, and therefor becomes imperative have a rectifier. The rectifier implemented was a textbook classic full-wave rectifier. In paralell with the output was palced the power factor correction capacitor followed by the bridge, formed by four low-drop diodes. Between the bridge and the 100F UC, a 1ohm resistor was placed to make current readings.

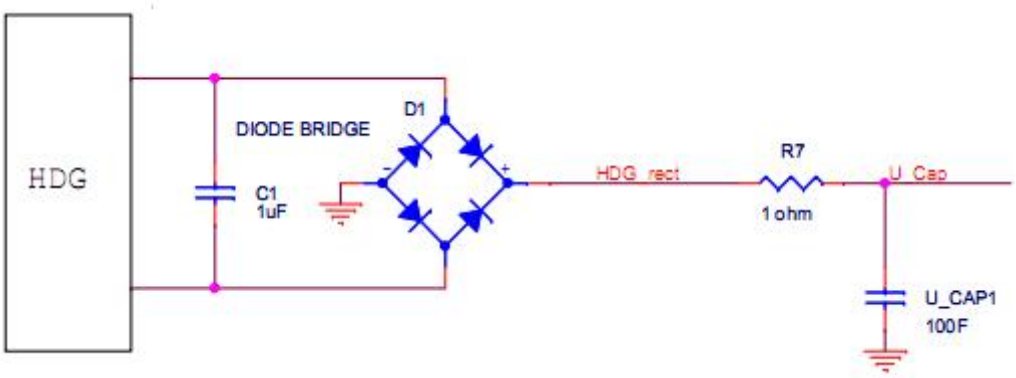


Figure 3.1: HDG and rectifying diode bridge

Before using the 1uF capacitor, the ouput was an almost perfect, noise free AC voltage signal. After placing the capacitor for power factor correction, the output changed in shape and peak voltage (which increased).

Figure 3.2: HDG output with paralell 1uF capacitor and no load

After connecting the remaining hardware (bridge, resistor and UC), the output changed to a square shaped signal.

Figure 3.3: HDG output with connected circuit

This is caused by the simple fact that the UC limits the peak voltage. This limitation is translated into current flowing from the HDG to the UC charging it. The current varies according with the SOC of the UC. The lower the UC's SOC, the lower the HDG's voltage, and the higher the charging current.

3.3 Microcontroller

The microcontroller is a simple PIC18F14K50. It was initially choosen due to its wide range supply voltage(1.8- 5V) and low-power features, which would be important for the electronic water meter. Pin count on this device (20) wasn't taken into account, and revealed insufficient, being later replaced by another model (18LF2520).

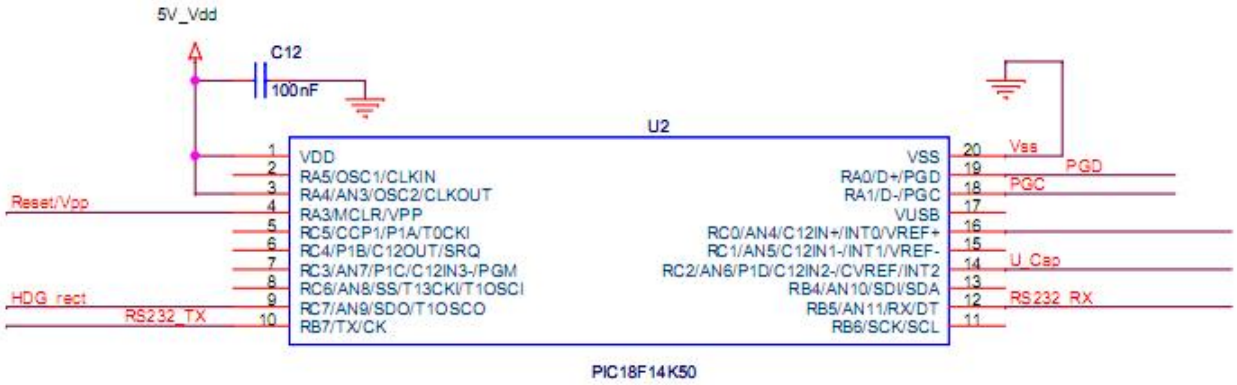


Figure 3.4: Microcontroller

It is supplied by a 5V voltage regulator (7508) and an external power source. The 5V DC is also used as the V+ ADC voltage reference.

Measures of voltage are taken before and after the 1ohm resistor (output of the diode bridge and UC) on analog channels 9 and 6 respectivelly. All the other data is determined based on these two measures. Furthermore, charge current is the voltage drop on the 1ohm resistor (V/R, R=1) and SOC of the UC (or energy level) is determined by the formula $\frac{1}{2} \times C \times V^2$, where V is the UC's voltage. These mathematical operations are performed on the computer. The software on the microcontroller has a very simple flow.

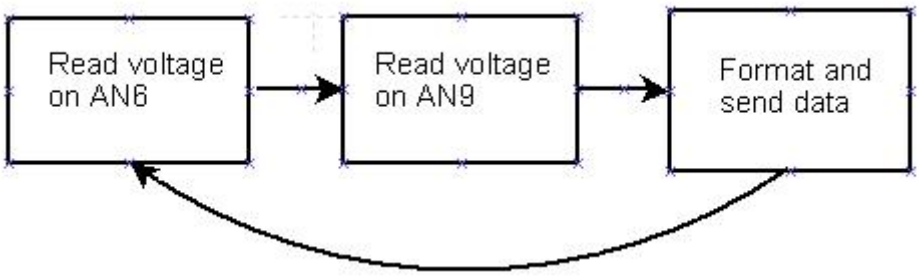


Figure 3.5: Software fluxogram

Voltage readings are taken on channels 6 and 9, formatted to a previously determined format and sent threw a serial communication protocol to a computer for processing.

The choosen format is a string formed by 8 characters. The first 4 characters are the voltage of the UC and the following 4 characters are the voltage at the output of the diode bridge. Both are in milivolt units.

3.4 Graphical user interface

The GUI for this system was developed in C#, using Visual Studio C# express. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Visual Studio's IDEhas embedded functions for serial port management, such as port opening, buffer checking and reading, among others. The final aspect of the developed GUI is shown in the next image.

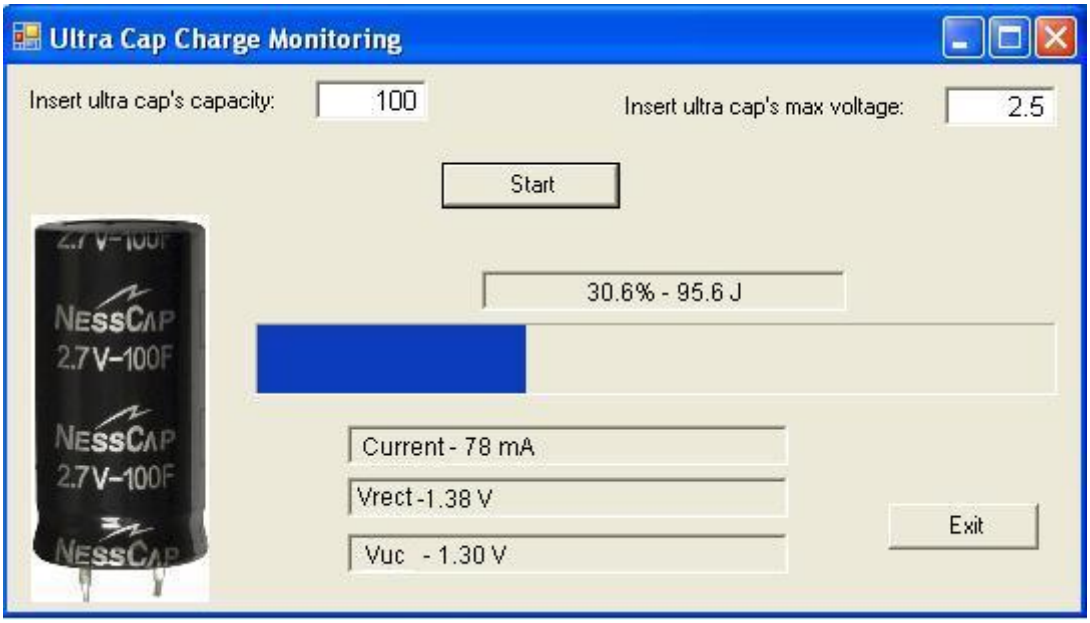


Figure 3.6: Graphical user interface

The user must insert the UC's capacity and maximum voltage, in order for the system to determine the SOC correctly. On the first bar below the Start button charge percentage and energy are presented, and the second bar is filled (from left to right) according with that percentage. The following three bars are used to display charge current, UC's voltage and bridge output voltage.

The fluxogram for this small software is as follows.

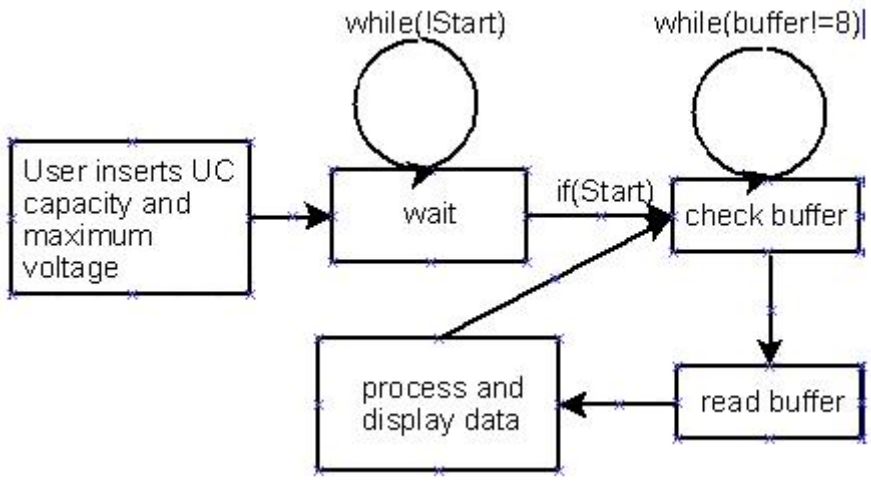


Figure 3.7: Graphical user interface software fluxogram

The user beings by entering values for the UC's capacity and maximum allowed voltage. The software will not read those fields while the user doesn't press the Start button. When

Start button is pressed, the values are read and the software beings a cycle where it checks if the buffer has the 8 expected bytes. When the buffer is filled with the 8 bytes, the data is read, processed and displayed as previously explained.

3.5 Conclusion

In this chapter was shown how the SOC of an UC can be determined. With simple hardware and software, an accurate SOC determination can be performed and displayed, in this case, in a graphical form. Another conclusion extracted is that the mathematical equation obtained on subsubsection 2.3.3.1 which relates the HDG's voltage with the water flow cannot be used. This dues to the fact that the voltage varies when the HDG is connected to a closed circuit, or in this case, to a rectifying diode bridge with an UC. For the water metering process, the frequency and its associated equation is the signal characteristic that must be used.

Chapter 4

WATER METER PROJECT - HARDWARE

4.1 Introduction

The electronic water meter is formed by four main hardware blocks:

- HDG, rectifying diode bridge and two UC (100F 2.7V);
- Microcontroller unit and surrounding electronic;
- LCD;
- GSM modem.

The HDG is the energy generator for all the electronics, and also the measurement instrument. It's a simple AC generator with an inductor (stator) and a magnet (rotor) that is spinned by the running water. Using the rectifying bridge, the output power of the HDG is used to charge two UC, which will be the only energy storage unit. The option of using two UCs instead of one was to extend the systems lifespan and avoid the use of step up converters which would drain the UCs rapidly. The choosen microcontroller was a PIC18LF2520 from Microchip. This model was chosen because of it's lowpower features (Microchip's nanoWatt technology) and wide operating voltage range (2 - 5.5V). This characteristic allows for the microcontroller to be connected directly to the terminals of the UCs without any voltage regulation, diminishing energy losses.

The LCD unit will be used for local readings. A button must be pressed by the user when he wishes to make a reading, and the system will display the data on the LCD for a period of about 5 seconds. This is considered to be enough for the user to memorize the reading and is almost irrelevant in terms of energy consumption.

For remote readings, a GSM module was placed, which sends an SMS with updated readings.

Both the LCD and GSM require different supply levels, 5V and 3.8V respectfully. This implies the introduction of two voltage regulators, to ensure proper device operation. Also, and once again because of the variable supply voltage of the microcontroller, a level shifter was needed to ensure the communication between the GSM module and the microcontroller using USART interface.

Over the next sections a more detailed description is done on each part of the system.

4.2 Energy storage

It was initially intended to use one single UC, but that would implie using voltage regulation. Using two UCs directly connected to the microcontroller (which is the only IC that is always active), we eliminate the need for voltage regulations. The chosen UCs are 100F 2.7V, that corresponds to one single 50F 5.4V UC (two equal capacitors in series are equivalent to one capacitor with have half the capacity). Fully charged we have a total of 729J of energy stored ($\frac{1}{2} \times 50 \times 5.4^2$). For the system to remain operational (except for the GSM module) and able to perform water metering, the minimum allowed voltage is 2V, which corresponds to the minimum supply voltage of the microcontroller. Therefore 100J of energy ($\frac{1}{2} \times 50 \times 2^2$) will remain stored in the UCs. This means that the storage system can transfer 86.3% of its energy to the electronic system (not accounting for self-discharge). The remaining 13.7% can't be transfered to the system with the chosen configuration (directly connected) and a boost converter is necessary to extract that remaining energy if necessary.

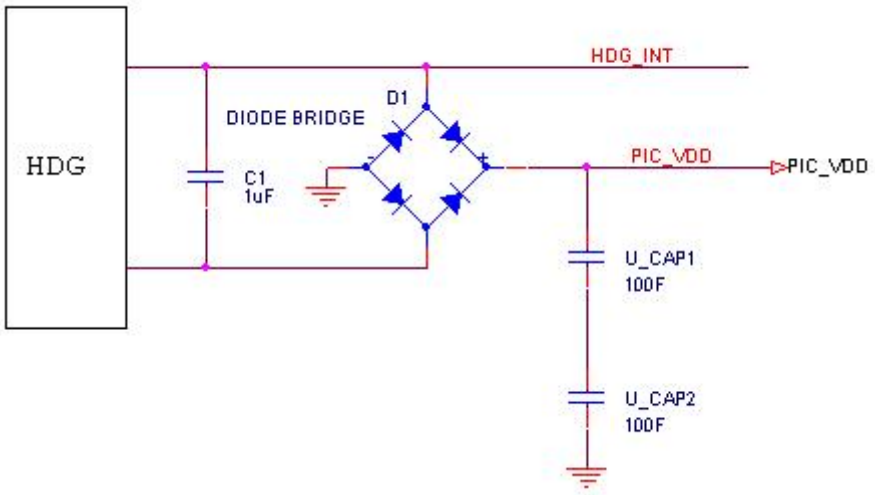


Figure 4.1: HDG, diode bridge and UCs schematic

Figure 4.1 shows the orcad schematic of the power source/storage block. As previously explained, the HDG's AC signal is rectified and charges two UCs. Due to the it's inductive characteristic, the output is boosted by a 1uF power factor correction capacitor. This value was obtained empirically and is detailed on section 2.3. The net PIC_VDD connects directly to the microcontroller and contains a DC voltage from the UCs. To measure the water flow rate we must use the AC signal directly from the HDG's output and measure it's frequency. So a connection was done between one of the outputs and a pin of the microcontroller. Details on this connection and measuring algorithm will be given at a later section.

It it common to use regulation on the UCs so they don't overcharge. In this case, no regulation is necessary due to the fact that the HDG has a peak voltage limited to around 11V. Therefore, the UCs voltage will never be much higher than 5.2V: 11V divided by two (from the rectifier) minus the drop on two diodes (around 0.15V each).

4.3 LCD

The LCD chosen was a standard 16x2 segments similar to the one on figure 4.2.

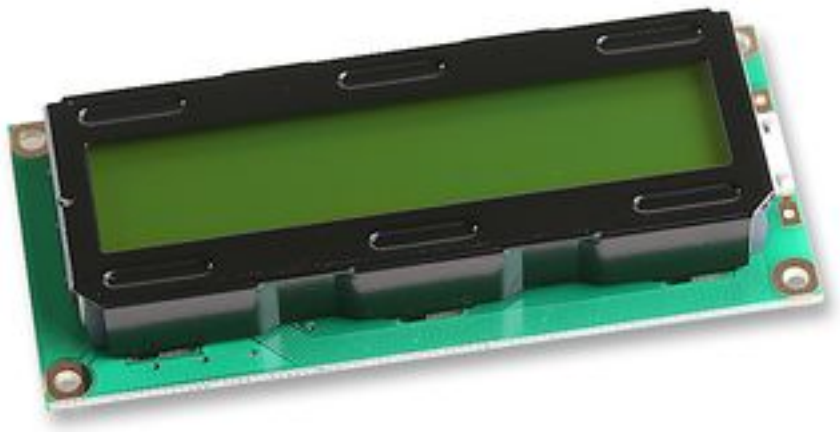


Figure 4.2: LCD

The typical supply voltage for the LCD is 5V, but can vary +/- 10% (from 4.5 to 5.5V)[28]. Since the power supply, hence, the UCs can vary from 5.2V to 2V, it is imperative to implement a regulator (with step up characteristics) to give the LCD a steady DC supply. For this porpuse, an IC from Texas Instruments was chosen (TPS61200), because it can regulate 5V even with low inputs (down to 0.5V)[29]. This regulator is only available in a QFN package which is very small and hard to mount[30]. A small PCB board was done to test the device but due to the components size it wasn't possible to make it work. So the alternative was to get an evaluation module for this component (TPS6120x-EVM) which solved this problem for the prototype development.

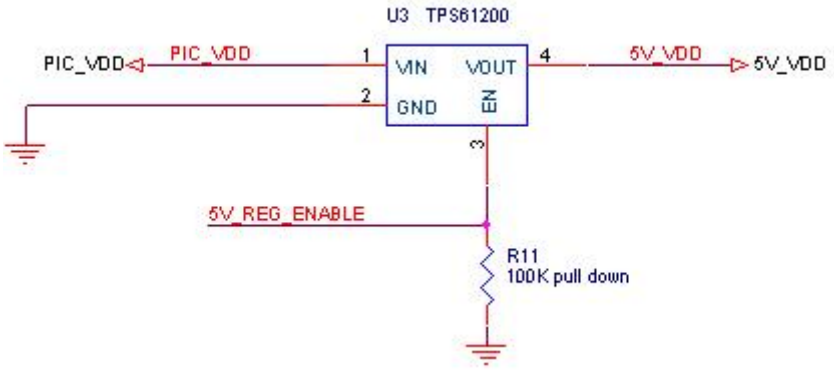


Figure 4.3: 5V Regulator

On the input of the regulator the voltage of the power source is applied to obtain a steady 5V regulated voltage on the output. In order to shutdown the device, a pull-down resistor was connected between enable pin and ground. The enable is active high, meaning that a signal equal to Vdd must be applied on this pin to turn on the regulator. Most of the time it is to remain off, only to be activated if the user requests a local reading, hence the LCD must be turned on.

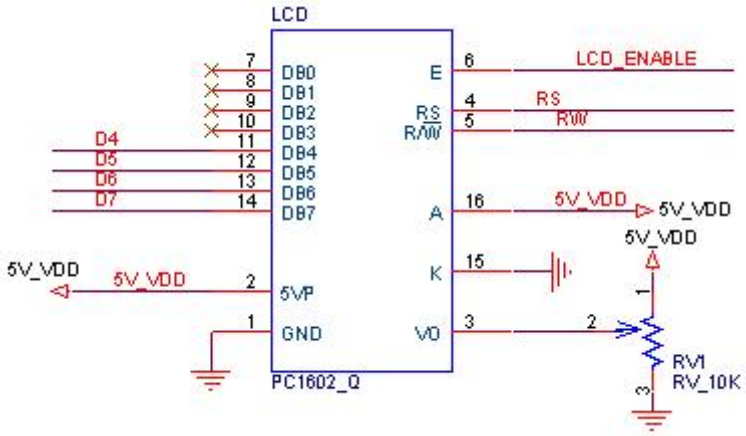


Figure 4.4: LCD schematic

On figure 4.4 the schematic of the LCD is shown. Pin 11 to 14 are data pins used to write and read data. Pin 4 is the register selection pin, indicating if we want to access instructions or data registers. Pin 5 is used to signal if the operation will be read (active high) or write (active low). Next, there is a pin to activate or deactivate the LCD (pin 6) which must be used in some situations. On pins 16 and 2 DC voltage supply is applied and Ground is connected to pins 1 and 15. To adjust contrast, a pontentiometer is connected between 5V and ground, with the middle point connected to pin 3. Pins 7 to 10 are not used in this case, since communication can be established between the LCD and the microcontroller by using only 4 data pins.

The LCD acknowledges high voltage levels starting from 0.7Vdd, hence, over 3.5V. If the UCs voltage drops below that value, it is possible that the LCD doesn't not work properly.

4.4 GSM modem

For remote data transmission, SMS texting seemed the wisest choice, since the user can be virtually anywhere on the planet and still receive an sms with meter readings. Other technologies were pondered such as bluetooth or zigbee, but were dropped out due to their limited range. The modem of option was a GC864 from Telit.

Figure 4.5: GC864 top view

Figure 4.6: GC864 bottom view

This modem must be supplied with 3.8V, and similiar to the LCD, a regulator was inserted in the circuit. When transmiting, the modem sends bursts at a base frequency of about 216Hz, where current peaks can be as high as 2A[31]. Because of this current demand, and as precaution, the regulator chosen was a TPS7580 from Texas Instruments. It is a 3A LDO regulator with adjustable output voltage[32]. This way the regulator operates in a voltage safe-zone, prevents overheating or high voltage drops during transmission. During the first tests with the modem, sometimes the modem was doing a reset every 4 or 5 seconds. Later it was observed that it only happened with the antena connected. The conclusion was obvious: when the modem had the antenna connected it tried to establish communication with the mobile network. In order to do so, it had to make burst transmission, which demanded high current peaks. Despite being able to respond to those current peaks, the regulator suffered a voltage drop that caused the modem to reset. The problem was only solved by placing a parallel 1000uF capacitor with the output of the regulator to prevent this voltage drop.

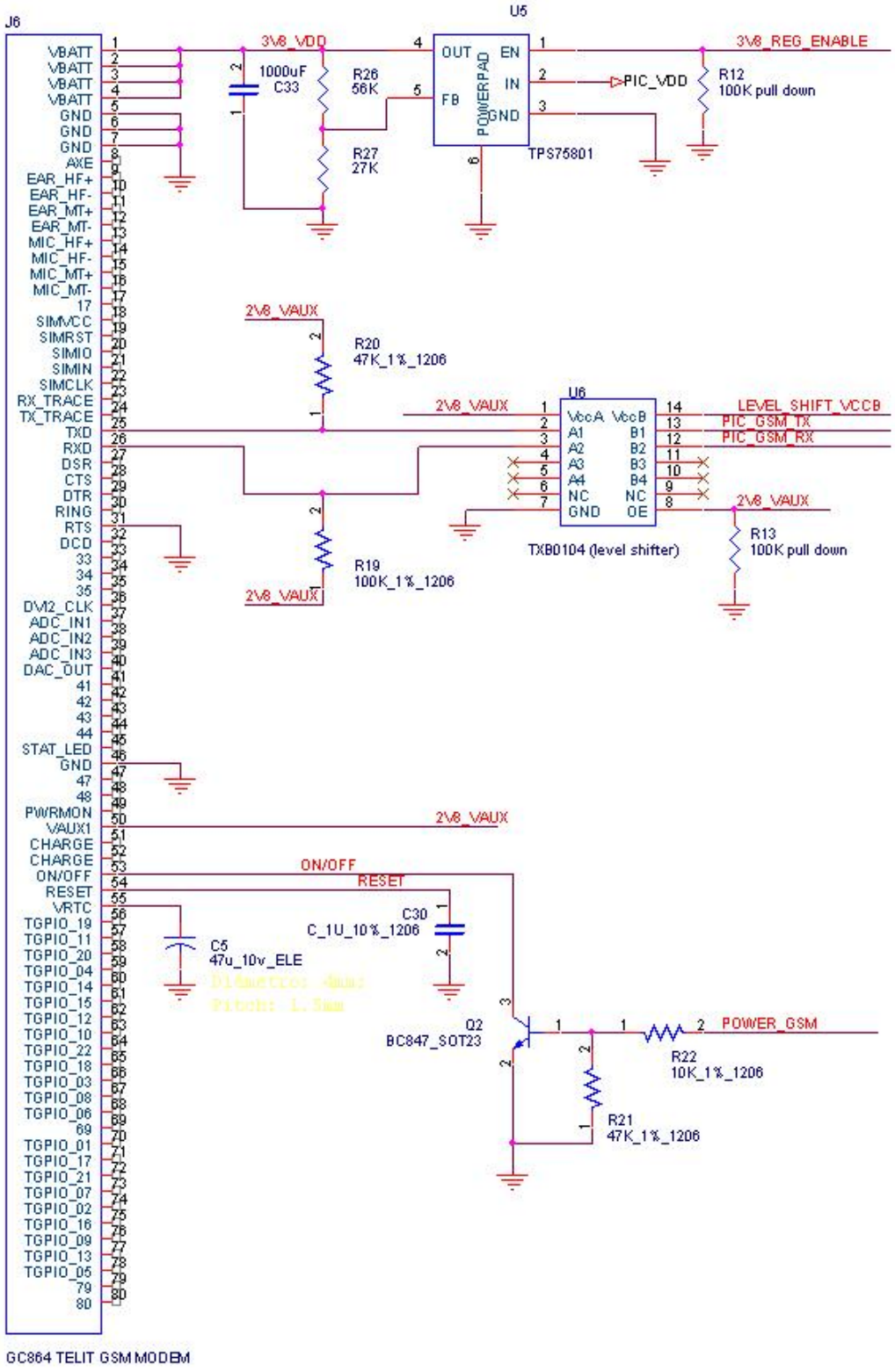


Figure 4.7: GSM modem schematic

The 3.8V regulator also has an enable pin, controlled by the microcontroller. This way we have total control over it, and it remains off during most of the time to ensure minimum current leaks. Also in this case, a pull down resistor was applied between the pin and ground.

Resistances R26 and R27 were calculated based on the information contained in the device data sheet. The normal procedure is to consider a value for R2 close to 30Kohm (we chose 27Kohm) and use the formula $R1 = \left(\frac{V_0}{V_{ref}} - 1\right) \times R2 \backsimeq 56Kohms$, where Vref = 1.224V.

Capacitors C5 and C30 were placed as a recomendation found in the modem's hardware guide to ensure a certain voltage level on those pins. The PNP BC847 transistor and resistors R21 and R11 are used to turn on and off the modem. To do so, the ON/OFF pin must be tied low for a certain period, as described in figure 4.8. When a voltage is applied at the base of the transistor, it drains current from the pin, causing the device to turn on or off.

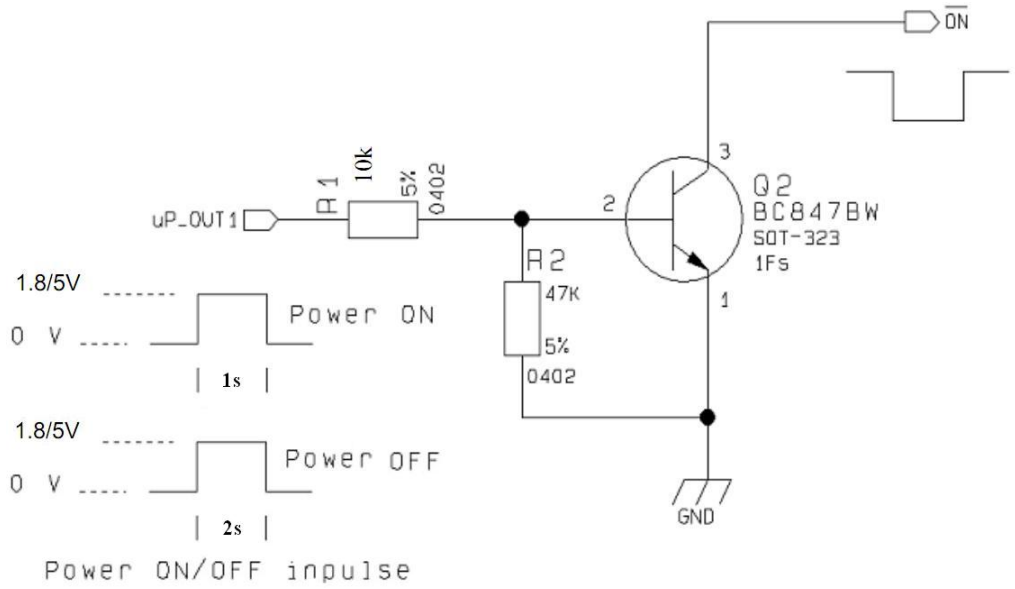


Figure 4.8: Modem ON/OFF

The modem communicates with the microcontroller using a USART interface. The input high level that the modem can operate with range from 2.1V (min) to 3.3V(max). Since the microcontroller can have a supply voltage above 3.3V, it was decided to place a level shifter, to translate signals from two different voltage levels. The chosen IC was TXB0104. It's a 4-bit bidirectional voltage level translator with automatic direction sensingfrom the last update of litrosMin variable, the calculation will be litrosMin=((freqAcumMed+160)/440)*(float)countSeg/[33]. Signals applied on port B are referenced to VccB and on port A, to VccA. VccA and VccB must be connected respectfully to 2V8 aux pin on the modem and a microcontroller pin that is set high before any transmission occurs. It can't be done otherwise, because VccA can't be higher than 3.6, and also has to be smaller or equal that VccB, which can go as high as 5.5V, aproximatelly the maximum theoretical voltage of the UCs.

This configuration ensures the comunication between the modem and the microcontroller, while protecting the modem TXD and RXD pins from over voltage.

For the level shifter to become active, both the OE and the VccB pins must be driven high by the microcontroller. Although they're high impedance pins, we must prevent any current leaks to prolong the system's energy autonomy. If the VccB pin was connected directly to the UCs, some current would get drained through the pin. Again, a pull down resistor ties the enable pin to the ground to ensure device shutdown, even if the correspondent microcontroller pin is on a tri-state.

4.5 Microcontroller

The microcontroller chosen for this project was a PIC18LF2520 from Microchip. As previously stated, it has a wide operating voltage range (2-5.5V) and a sleep mode where the current can drop to less than 1uA[34]. It will remain in this mode most of the time, and will be activated by external interrupts such as the HDG signal indicating that water started to flow, or the local reading button, indicating that data must be shown in the LCD display. The microcontroller has an internal oscillator, but it can have an error as high as 1%. Therefore, an external 4Mhz crystal is used. This frequency was chosen because it still allows the microcontroller to have a voltage supply spectrum of 2-5.5V, and also, to have a USART baud rate of 115200Kbps, which is the default baud rate of the GSM modem.

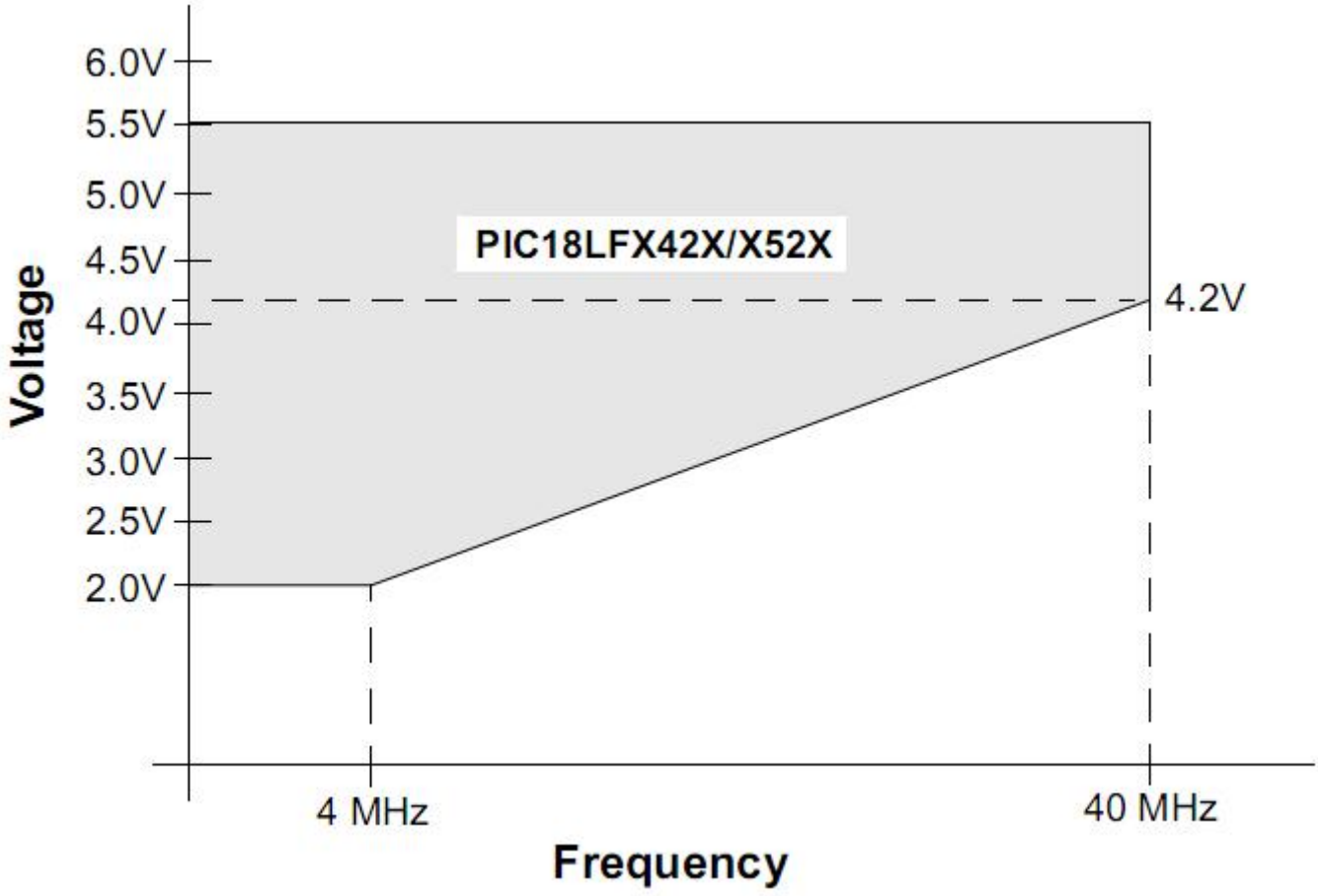


Figure 4.9: Voltage Vs Frequency

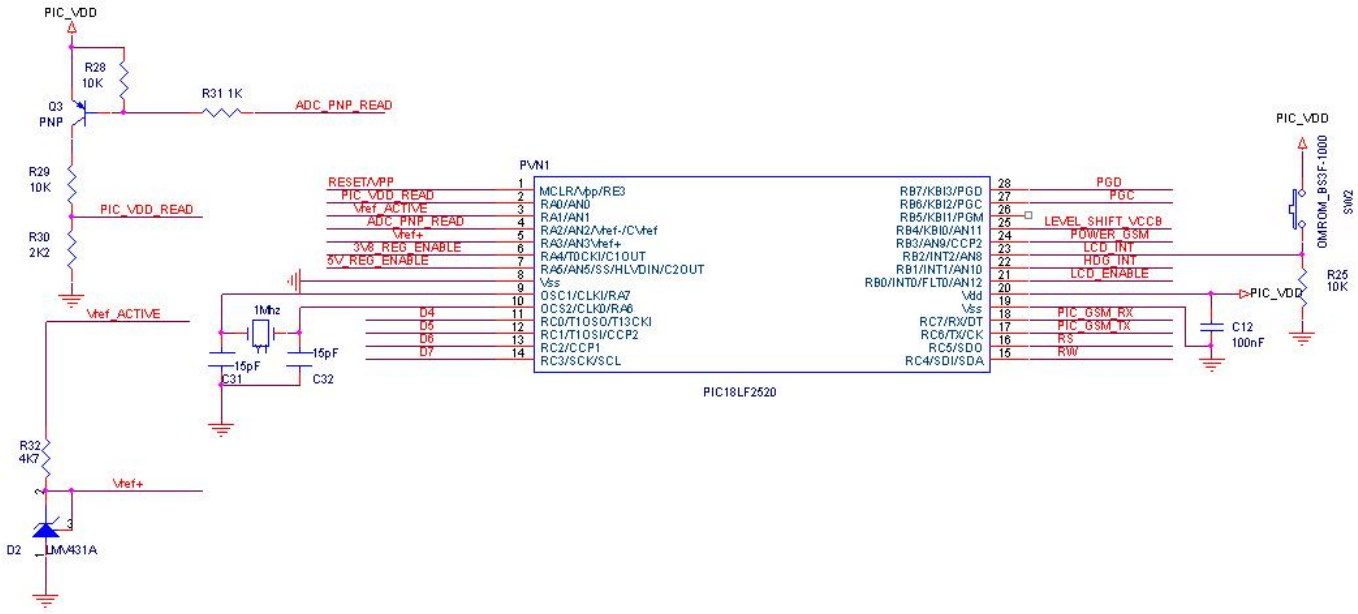


Figure 4.10: Microcontroller and attached electronics schematic

This device has an internal oscillator with variable and programmable frequency, but it can have an error of 1%, which would affect the measures of the water flow, since they're time-based. To ensure maximum precision, an external 4Mhz crystal was placed as the device's clock source. This frequency was chosen because it allows for the device to have USART communication with 115200 baud-rate, which is the GSM modem's default baud rate. Also, as detailed on the electrical characteristics of the microcontroller, it is the maximum frequency for which the device is still capable of operating from 2.0V to 5.5V[34]. As a drawback, it will drain more current during operation, but not during sleep mode. Pin 23 is an external interrupt pin, and causes the microcontroller to wake up from sleep mode. To do so, it was tied low using a 10K resistor connected to the ground, and a buttn switch connected to the supply voltage. When the user wants to make a local reading, the button is pressed, and a high voltage level appears on the pin, causing the microcontroller to wake up and start the associated routines that will read the data and send it to the display. After the reading period, the LCD and voltage regulator are turned off and the microcontroller goes back to sleep mode.

One of the conditions for the system to send an sms is that the UCs voltage must be over 4V. This ensures that the voltage regulator will work properly and that there is enough energy stored to make a successful transmission. The problem was how to read the UCs voltage level without a steady voltage reference, and also, without draining extra current. The solution for this issue was based on a simple PNP transistor and a LMV431 shunt regulator (LMV431). The shunt regulator is a 3 termiigh to low nal device that has a Vref of 1.25 V when biased. So, to bias the device, it was connected to a microcontroller pin through a 4K7 resistor, and the refence pin was connected to the cathode and to the external Vref+ pin of the microcontroller. To have this external voltage refence, the only thing that must be done is active high the Vref_ACTIVE pin, and a 1.25V voltage reference will appear on Vref+ pin. This only takes a few microseconds, so current leakage is insignificant. If no voltage is applied on Vref_active, the shunt regulator isn't biased, and hence, no current will be driven.

As for the transistor, the base is connecteuled to a 1K ohm resistor, which in turn is connected to a microcontroller pin (pin 4 - ADC_PNP_READ). The emitter is connected to the microcontroller's supply, hence, the UCs V+. As for the collector, connects to a resistor

voltage divider. As long as pin 4 is active high, the transistor is cut-off and there's no current flowing. When the pin drops to a low level, the transistor becomes saturated, and the supply voltage can be determined. The analog reading is done with 8 bit resolution (256 levels) where 0 is 0V and 255 is 1.24V, the external voltage reference. To have more accurate results, 10 readings are done and the final result is the average of those readings. To calculate the supply voltage, the following formula is applied $Vcc = Vread \times \frac{12.2}{2.2} + Vce$. Vce is the emitter-colector voltage drop and is aproximatelly 0.2V. The goal is not having a precise measurulee but an indication that it's "safe" to send an sms without expecting power loss.

As for other connections, pins 6 and 7 are used to activate/deactivate the voltage regulators. Pins 9 and 10 are the crystal's input and pins 11 to 14 are data pins, used communicate with the LCD. Pins 15, 16, 21 and 25 are also used to control the LCD and their functions have been described previously. Pins 17 and 18 are the USART interface pins and are used to communicate with the GSM modem. Pin 24 is used to turn the modem on and off as shown previsouly. Pins 27 and 28 are used to program the microcontroller with the proper firmware.

4.6 PCB

The PCB developed is shown on the following figures. The first one shows only the PCB, and the second shows the PCB with the GSM antenna and the hydrogenerator used for simulation and testing.

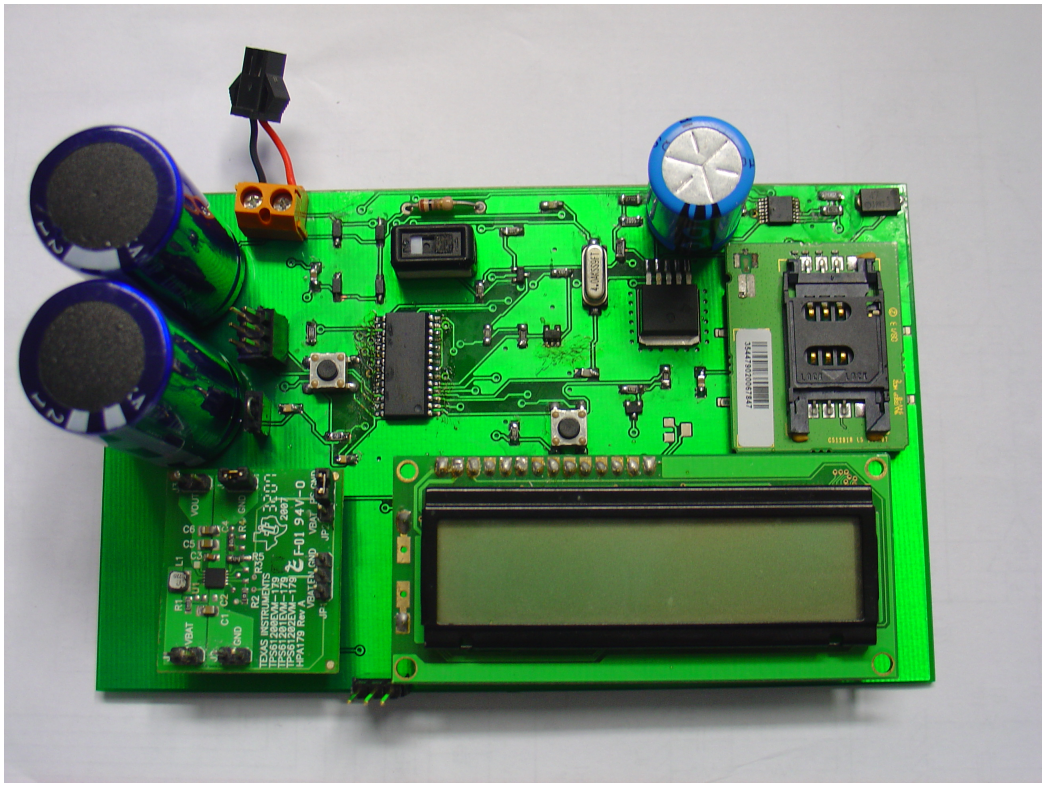


Figure 4.11: PCB

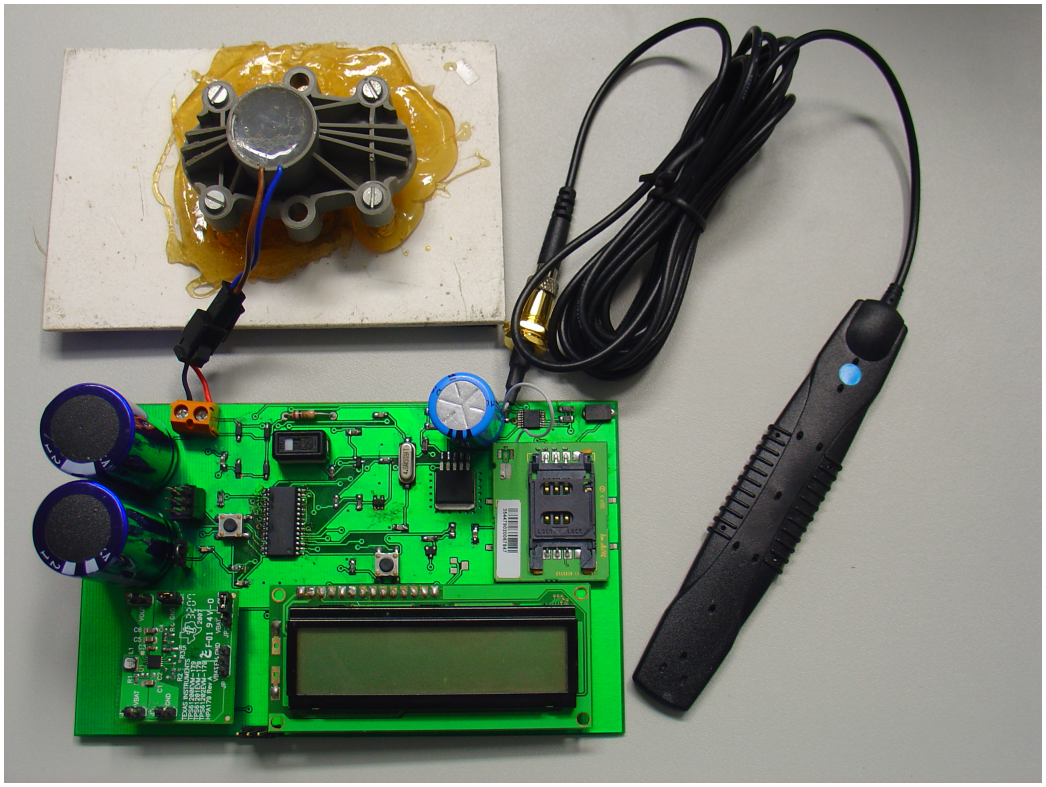


Figure 4.12: PBC, antenna and hydrogenerator

4.7 Conclusion

The hardware was chosen and developed in such way that allowed for maximum energetic saving and performance. Most of the chosen IC have enable/disable pins, to avoid being constantly in operational mode. This way, their lifespan is also prolonged. The initial chosen microcontroller was the same one used in the charging system (PIC18F14K50) but had an insufficient pin number. The biggest challenge was te get the GSM module operational, due to the problems caused by the current peaks it required.

Chapter 5

WATER METER PROJECT - SOFTWARE

5.1 Meetering process

As a reminder, one of the external interrupt pins of the microcontroller (pin 22 - INT1) was connected to the output of the HDG, where the signal is shaped as shown on figure 3.3. It is configurable if the interrupt will occur on a low to high transition or otherwise. The correct option is high to low, because when the HDG is not working, there is a positive voltage on that point of the circuit that could cause the microcontroller to be constantly on the interrupt routine. This way, it's made sure that the interruption only occurs when the water starts flowing.

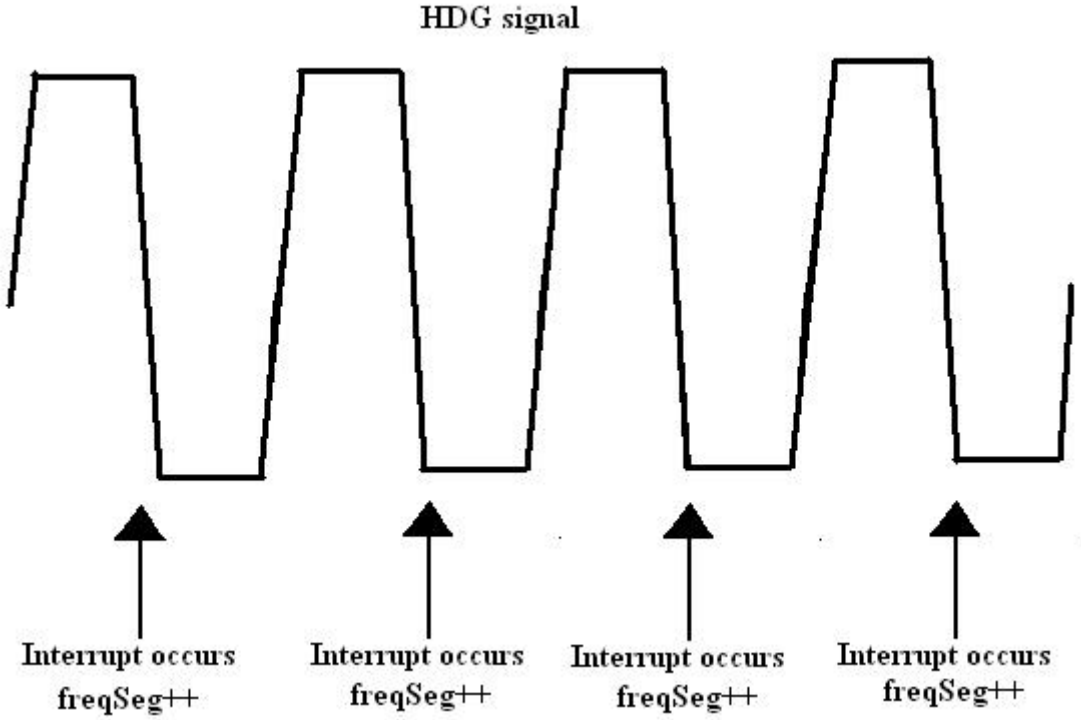


Figure 5.1: HDG's interrupt process

To make the reading, the chosen approach was to count the number of interrupts during each second, and hence, the frequency. These frequency values are used to obtain an average frequency at the end of each minute. The average will then be applied on the formula obtained on section 2.3 which relates the water flow with the signal frequency. This will give us the amount of water that passed through HDG on each minute. A variable called litrosMin is updated with the total number of liters at the end of each minute. One other variable called litrosTotal makes the sum of all the updates on litrosMin, while the HDG is running. When it stops running, the detection is made with a timer that overflows after 65ms without an interruption on INT2. If an interruption caused by the HDG to be apart from the previous one more than 65ms, meas that it was running on 15Hz. From the study of the hydrogenerator it is known that it only starts working with a minimum water flow of 0.8 L/min, which corresponds to around 190Hz. It is considered that water stopped running when the timer overflows. When the HDG stops, the average frequency is done considering the number of seconds passed. If, for instance, 38 seconds passed from the last updatederived of litrosMin variable, the calculation will be litrosMin=((freqAcumMed+160)/440)*countSeg/60. freqAcumMed is the average frequency up to that point.

Every time the water stops flowing, the previous counting is read from the EEPROM memory of the microcontroller, added with the consumption and saved again into the memory. 2 bytes of the EEPROM memory are used for this. When the value reaches 1000 liters it is considered that 1 cubic meter has been consumed and it is added to the consumption in cubic meters. This value is also saved to the EEPROM memory and the liter counting is reset back to zero.

5.2 Functions overview

There were developed functions to serve several porpuses. There are two interrupt level functions, high and low. On low priority are the external interrupts caused by the HDG and LCD reading request. High priority are the timer's overflow interrupt.

Four functions are associated with the voltage regulators:

- *void enable5V(void)* - enables the 5V regulator by activating high the corresponding pin;
- *void disable5V(void)* - disables the 5V regulator by activating low the corresponding pin;
- *void enable3V8(void)* - enables the 3.8V regulator by activating high the corresponding pin;
- *void disable3V8(void)* - disables the 3.8V regulator by activating low the corresponding pin.

The function *float readUltraCapVdd(void)* returns the value of the supply voltage to decide if an sms should be sent or not. *void goSleep(void)* orders the microcontroller to enter sleep mode, disabling most of its peripherics and saving energy from the UCs.

Two functions are associated with the EEPROM management:

- *void eepromWrite(unsigned char addr,unsigned char eeData)* - writes one byte of data (eedata) to a selected memory address (addr);
- *unsigned char eepromRead(unsigned char addr)* - reads one byte of data from a single memory address (addr).

A separate library called *display_cmd_nibble.h* contains functions to operate with the LCD. The two most revelant functions used are void displayInit(void), which initiates and configures the display, and *void lcdPrint(unsigned char line, unsigned char col, const char *data)*, that writes into the display the characters contained in the array *data* starting at line *line* and column *col.*

Another separate library is used to communicate with the GSM module. This library is called *myGSM.h* and follows a description of its most important functions:

- *void GsmOn(void)* - activates the transistor shown on the previous chapter that is used to turn the GSM modem ON;
- *void init_gsm(void)* - initiates the modem and writes the parameters for sms;
- *void envio_sms(const char *texto)* - sends an sms containing the string texto.

5.3 System's general fluxogram

As previously stated, most of the time the water meter will be in sleep mode. Only two events can wake it from sleep mode: water flowing and LCD reading request. The supply voltage will be read each time water stops flowing. It doesn't make sense to read it at any other time, because the energy level on the UCs will only rise if water flows through the HDG. In case this voltage is below the treshold level, it will go back to sleep mode. If it is over the treshold level, and sms with the totalized counting will be send to the user. If the LCD reading button is pressed, the microcontroller will wake up from sleep mode, read the data from its EEPROM and send it to the display which will show the data for a period of 5 seconds. After this period, the display will be shut down and the microcontroller will enter sleep mode.

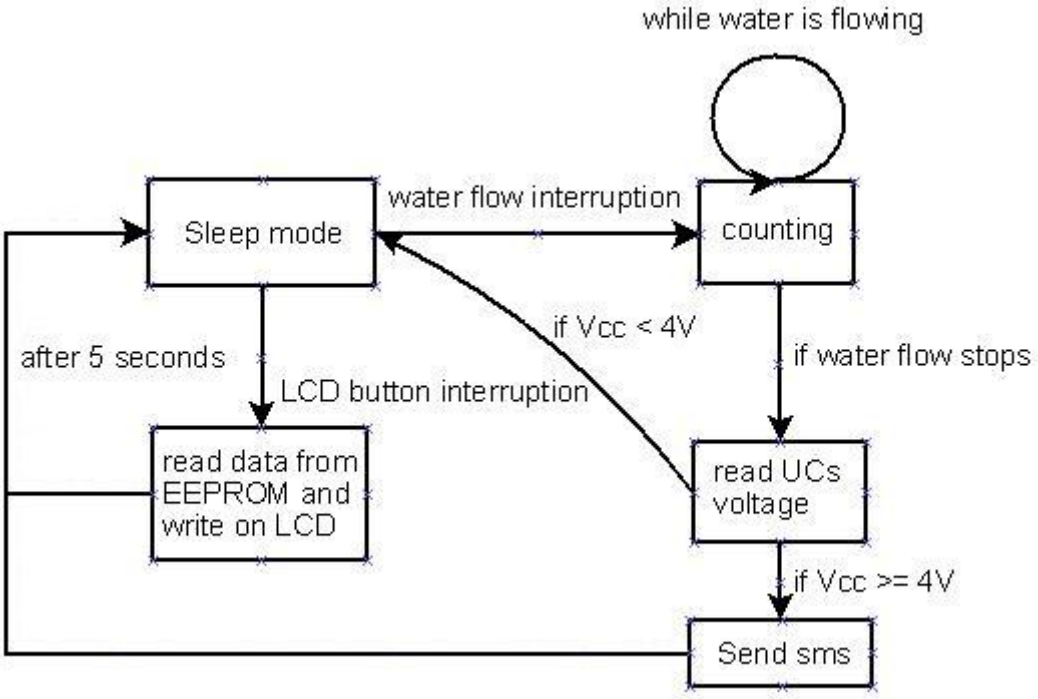


Figure 5.2: Fluxogram

5.4 Conclusion

Just like the hardware, software was developed thinking in terms of energy saving. The microcontroller is always in sleep mode, except when the water flows or the user wants to make a reading. The drawback of this system is that it will always send a message after a period of flowing water (if Vcc>=4.2V). This means that, if the UC are charged above 4.2V and the water is turned on 5 times in one day, 5 sms will be sent. With a calendar control it would be possible to program the sms with a specifical mininum interval (e.g. two or four weeks).

Chapter 6

Conclusions

6.1 Final system analysis

The final system works as expected: the LCD displays the correct results, the metering is done accordingly an the GSM module sends the data rapidly and spending very little energy. Its consumption varies between less than 10uA and 20uA, possibly because of temperature drifts. It is hard to make an estimation of how long the water meter will be operational because it depends on the UC SOC and on the average current consumption which can vary a lot. The main objective of having an energetically self-sustained system was also accomplished. It should be noticed that for the meter to work properly, it should be placed in an area with a strong signal level, with the penalty of not being able to send data over GSM.

Let us now take a look to the prototype final cost, accounting only for the materials and components used, and excluding the engineer's work.

- PCB - 40€
- GSM - 50€
- LCD - 6€
- Microcontroller - 5€
- 5V Regulator - 4€
- 3.8V Regulator - 6€
- Level Shifter - 2€
- UC - 2€ each
- Other components - 4€
- Hydrogenerator - unknown cost

This brings to a total of 121€ for a prototype, excluding the HDG, which is property of Vulcano and can only be bought as a integrating part of the Click HDG Water Heater model. Also, the component prices presented are for single parts, not quantity prices. For a production, the total cost could easily drop under 50€, accounting only for the electronics.

In terms of energy autonomy, with full charge and no water flow, the system will remain in an active zone (UC voltage above 2V) for a minimum period of one to two weeks. This period can vary because of the different current consumption which is temperature dependent.

6.2 Future work

To extend this project even further, the following ideas are proposed:

- compare the performance of the water meter prototype with a pattern, using it to make an error analysis and correction;
- implement a calendar control system for the SMS (using, for instance, an RTC);
- develop a database to collect and process the data from the water meters;
- improve the charging method by developing a MPPT of the HDG;
- implement error control, such as signaling the user when the GSM modem fails or other related errors.

Bibliography

[1] P. Zane Satterfield and e. s. Vipin Bhardwaj, *Water Meters Tech Brief*.

[2] *consulted in September 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Water_meter

[3] *consulted in September 2009*. [Online]. Available: http://www.ehow.com

[4] *consulted in October 2009*. [Online]. Available: http://www.engineersedge.com/instrumentation/nutating_disk_displacment_meter.htm

[5] *consulted in October 2009*. [Online]. Available: http://www.smartmeasurement.com/en/documents/flowmeter/PDOscillatingPiston.asp

[6] *consulted in October 2009*. [Online]. Available: http://sensors-transducers.globalspec.com/LearnMore/Sensors_Transducers_Detectors/Flow_Sensing/Turbine_Flo

[7] *consulted in October 2009*. [Online]. Available: http://www.flowconditioner.com/articles/orifice-meter.html

[8] *consulted in October 2009*. [Online]. Available: http://www.omega.com/prodinfo/ultrasonicflowmeters.html

[9] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Magnetic_flow_meter

[10] *consulted in September 2009*.

[11] *consulted in September 2009*. [Online]. Available: http://cityroom.blogs.nytimes.com/2009/03/24/city-turns-to-wireless-for-water-bills/

[12] *consulted in September 2009*. [Online]. Available: http://houston.bizjournals.com/houston/stories/2009/09/07/focus2.html

[13] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Battery_(electricity)

[14] G. Lagunoff, *Automotive Hydrib Technology Status Function and Development Tools*, Lulea University of Technology.

[15] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Lead-acid_battery

[16] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Lithium-ion_battery

[17] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Nickel-metał_hydride_battery

[18] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Nickel-cadmium_battery

[19] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/D_battery

[20] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Electric_double-layer_capacitor

[21] *consulted in January 2009*. [Online]. Available: http://www.ultracapacitor.net/whatareultracapacitors.php

[22] *consulted in October 2009*. [Online]. Available: http://lees.mit.edu/lees/ultracapacitors.htm

[23] *consulted in January 2009*. [Online]. Available: http://www.groept.be/www/dam/HYDROpower.htm

[24] *consulted in January 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Hydroelectricity

[25] *consulted in January 2009*. [Online]. Available: http://en.wikipedia.org/wiki/Three_Gorges_Dam

[26] *consulted in January 2009*. [Online]. Available: http://www.investinportugal.pt/MCMSAPI/HomePage/BusinessActivities/ElectricAndElectronic/Vulcano

[27] *consulted in January 2009*. [Online]. Available: http://www.vulcano.pt/index.php?article=2&visual=1&artid=2&tema=8

[28] *LCD POWERTIP 1602 Specifications*.

[29] *TPS61200 Data Sheet*.

[30] *consulted in October 2009*. [Online]. Available: http://en.wikipedia.org/wiki/QFN

[31] *Telit GC864 Hardware user guide*.

[32] *TPS75801 Data Sheet*.

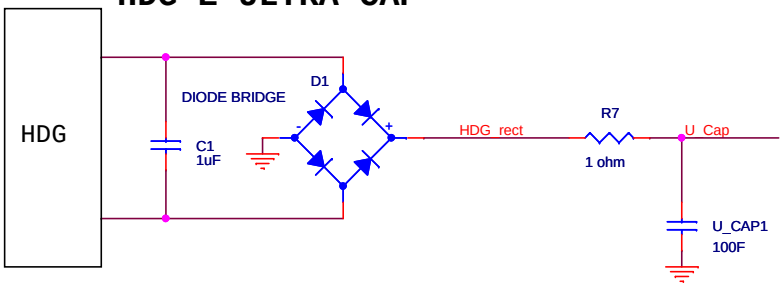
[33] *TXB0104 Level Shifter Data Sheet*.

[34] *PIC18F2420/2520/4420/4520 Data Sheet*.

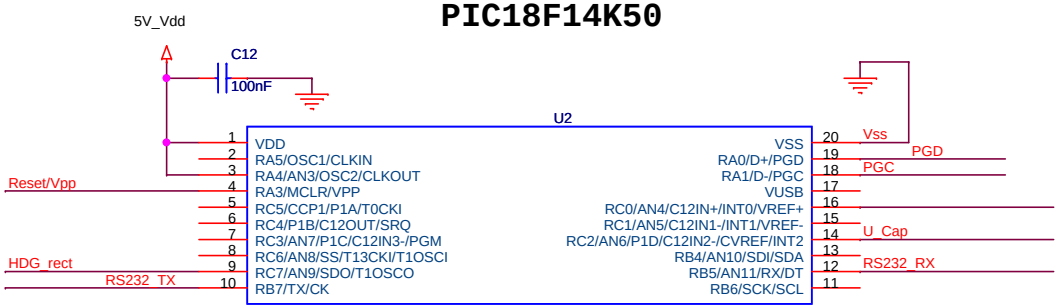
Apendix A

Hardware schematics

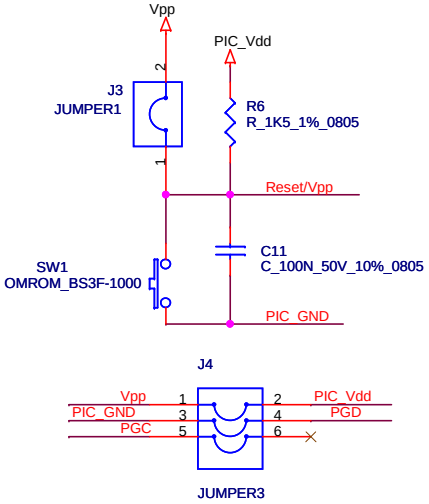
HDG E ULTRA CAP



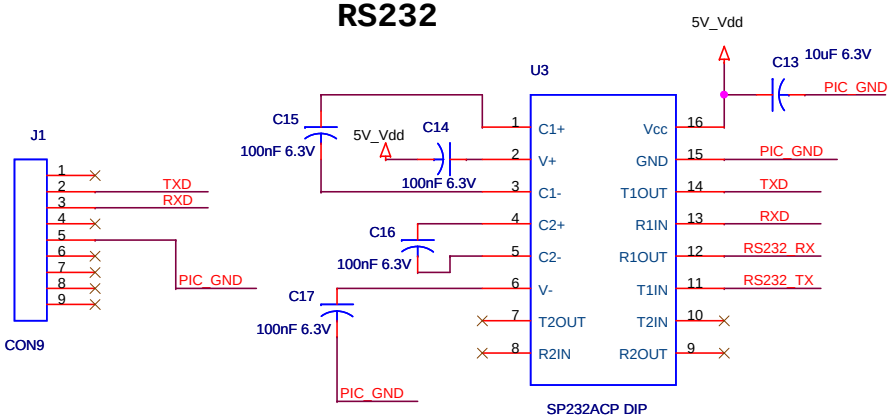
PIC18F14K50



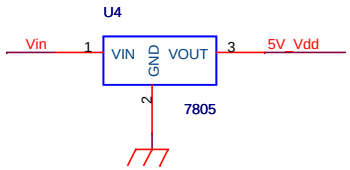
ICSP E RESET



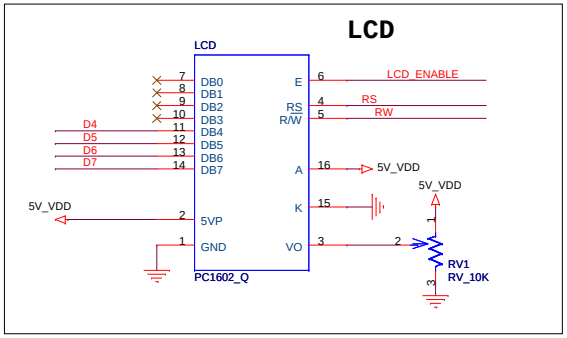
RS232

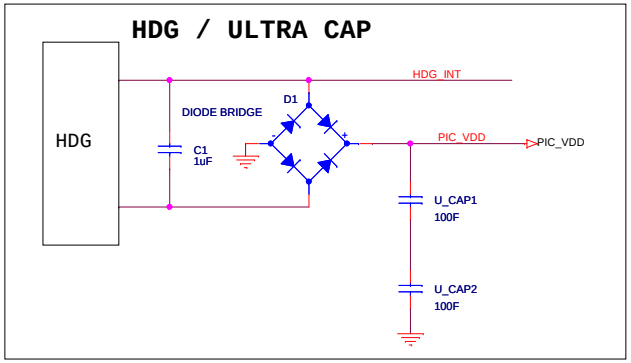


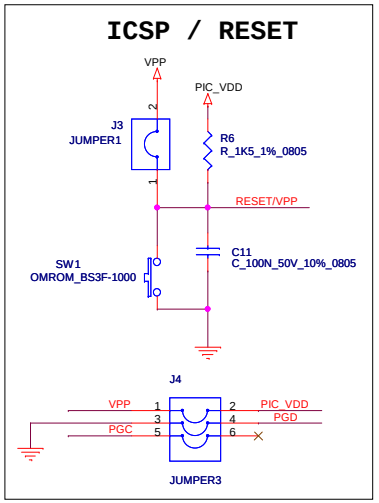
Voltage Regulator

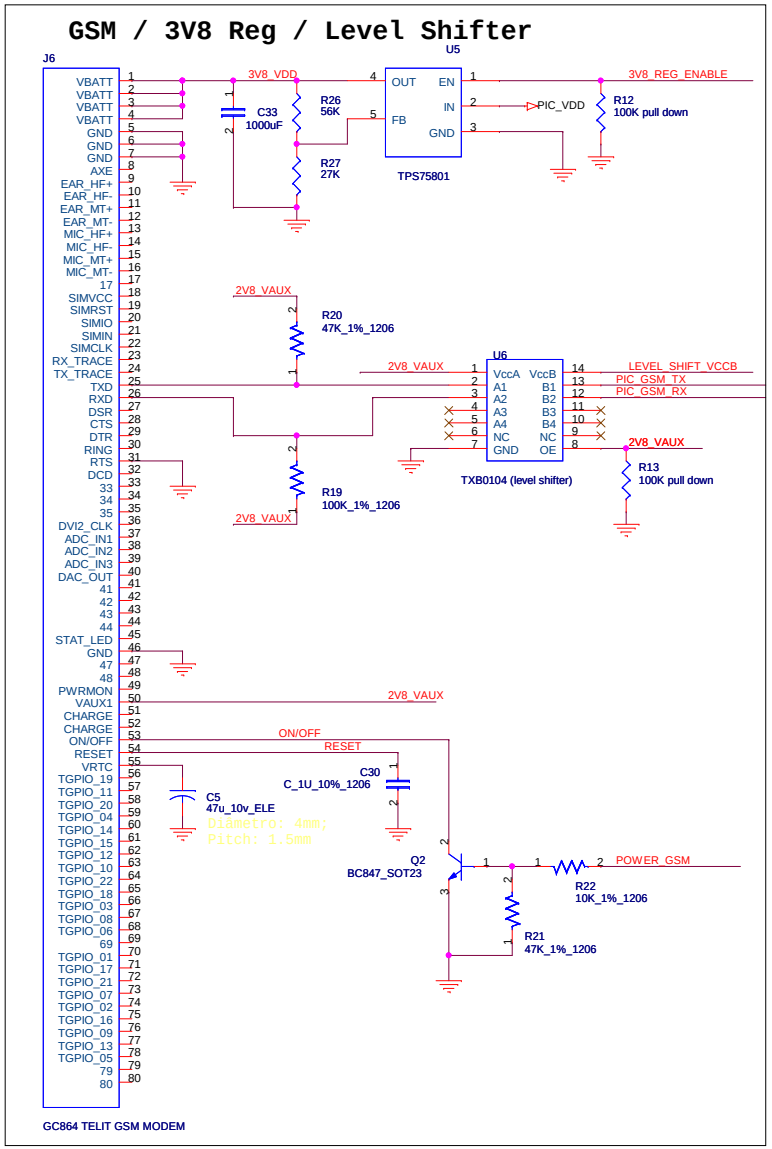


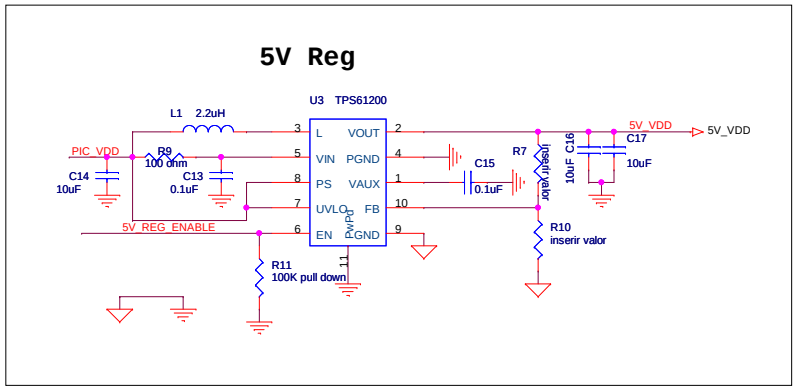
Title: Ultra Cap charging and monitoring		
Size: Custom<Doc>	Document Number	Rev: <RevCode>
Date: Friday, June 19, 2009	Sheet 1 of 1	

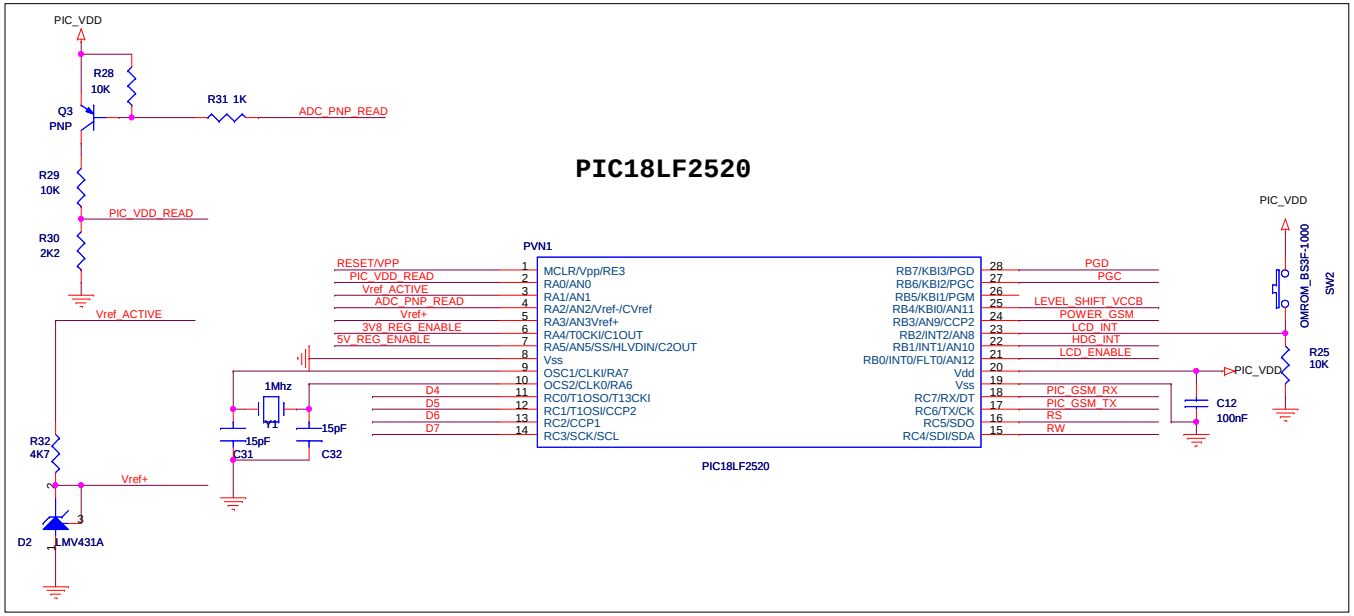












Title	Electronic Water Meter		
Size	Document Number		Rev
	Custom<Doc>		<RevCode>
Date:	Tuesday, July 28, 2009	Sheet 1 of 1	

Apendix B

Software – microcontroller SOC determination

```
#include <p18f14k50.h>
#include <stdio.h>
#include <delays.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <usart.h>


#define U_CAP_CAPACITY 100.0
#define U_CAP_MAX_V 2.5

void init_pic(void);
float ultra_cap_v_sense(void);
void energia (float volt_act, float volt_max, float cap, float* temp);
float hdg_rect_v_sense(void);
float current_calc(float v_plus, float v_minus);

void init_pic(void)
{
    /*clock source and speed*/
    //OSCCONbits.IRCF2 = 0; /*1 Mhz*/
    //OSCCONbits.IRCF1 = 1; /*1 Mhz*/
    //OSCCONbits.IRCF0 = 1; /*1 Mhz*/
    //OSCCONbits.IOFS = 1; /*frequency is stable*/
    OSCCONbits.SCS1 = 0; /*internal oscillator block*/
    OSCCONbits.SCS0 = 0; /*internal oscillator block*/

    /*port config*/
    TRISB = 0b10100000;
    /*RB5 RS232_RX
      RB7 RS232_TX*/
    TRISC = 0b10000100;
    /* RC2 U_Cap_Voltage AN6*/
    /* RC7 HDG_Rect AN9 */
    WPUB = 0b00000000; /*pull up disabled*/

    ANSELbits.ANS6 = 1; /*AN6 RC2 is analog input*/
    ANSELHbits.ANS11 = 0; /*RB5 digital input enabled*/
    ANSELHbits.ANS9 = 1; /*AN9 RC7 is analog input*/

   /*UART config*/
    TXSTAbits.TX9 = 0; /*8 bit transmission*/
    TXSTAbits.TXEN = 1; /*transmission enable*/
    TXSTAbits.SYNC = 0; /*asynchronous mode*/
    TXSTAbits.BRGH = 1; /*low speed*/
```

```
    RCSTAbits.SPEN = 1; /*serial port enable*/
    RCSTAbits.RX9 = 0; /*8 bit transmission*/
    RCSTAbits.CREN = 1; /*receiver enabled*/

    BAUDCONbits.BRG16 = 0; /*8 bit baud rat generator*/

    SPBRG = 25; /*Baud Rate 2400*/

    PIE1bits.RCIE = 1; /*reception interrupt enabled*/
    INTCONbits.GIE = 1; /*enables interrupts*/
    INTCONbits.PEIE = 1; /*enables peripheral interrupts*/
    RCONbits.IPEN = 1; /*enable priority levels*/

    /* ADC config*/
    ADCON1bits.PVCFG0 = 0; /*Positive reference supplied internally by Vdd*/
    ADCON1bits.PVCFG1 = 0; /*Positive reference supplied internally by Vdd*/
    ADCON1bits.NVCFG0 = 0; /*Negative reference supplied internally by Vss*/
    ADCON1bits.NVCFG1 = 0; /*Negative reference supplied internally by Vss*/
    ADCON2bits.ADFM = 0; //left justified
    ADCON2bits.ACQT0 = 1; //20 TAD
    ADCON2bits.ACQT1 = 1; //20 TAD
    ADCON2bits.ACQT2 = 1; //20 TAD
    ADCON2bits.ADCS0 = 0; //conversion clock Fosc/64
    ADCON2bits.ADCS1 = 1; //conversion clock Fosc/64
    ADCON2bits.ADCS2 = 1; //conversion clock Fosc/64

}

void energia (float volt_act, float volt_max, float cap, float* temp)
{
    float e_total, e_parcial, percent;

    temp[0] = 0.5*cap*volt_max*volt_max;
    temp[1] = 0.5*cap*volt_act*volt_act;
    temp[2] = temp[1]*100/temp[0];
}

float ultra_cap_v_sense(void)
{
    //Faz-se 10 leituras e tira-se a média
    float u_cap[10], med_v = 0.0;
    int i;

    ADCON0bits.CHS3 = 0; /*ADC channel 6 AN6*/
    ADCON0bits.CHS2 = 1;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS0 = 0;
    ADCON0bits.ADON = 1;//ligar ADC
```

```
    for(i=0;i<=9;i++)
    {
        ADCON0bits.GO = 1;//iniciar a conversão
        while(ADCON0bits.GO);

        //para fonte externa: alimentaçao 5V
        //u_cap[i] = (float)ADRESH*5/256;
        //para step up: alimentação 3.3V
        u_cap[i] = (float)ADRESH*3.3/256;
    }

    ADCON0bits.ADON = 0;

    for(i=0;i<=9;i++)
    {
        med_v = med_v + u_cap[i];
    }
    return (med_v/10);
}

float hdg_rect_v_sense(void)
{
    float hdg_v[10], med_v = 0.0;
    int i;

    ADCON0bits.CHS3 = 1; /*ADC channel 9 AN9*/
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS0 = 1;
    ADCON0bits.ADON = 1;//ligar ADC

    for(i=0;i<=9;i++)
    {
        ADCON0bits.GO = 1;//iniciar a conversão
        while(ADCON0bits.GO);
        //para fonte externa: alimentaçao 5V
        //hdg_v[i] = (float)ADRESH*5/256;
        //para step up: alimentação 3.3V
        hdg_v[i] = (float)ADRESH*3.3/256;

    }

    ADCON0bits.ADON = 0;

    for(i=0;i<=9;i++)
    {
        med_v = med_v + hdg_v[i];
    }
    return (med_v/10);
}
```

```
float current_calc(float v_plus, float v_minus)
{
    if(v_plus>=v_minus)
    {
        return (v_plus - v_minus);
    }
    else
    {
        return(v_minus - v_plus);
    }
}

void main (void)
{
    float u_cap_v, hdg_rect_v, current;
    float u_cap_data[3];
    unsigned int u_cap_v_send, hdg_rect_v_send;

    init_pic();
    PORTBbits.RB6=1;
    while(1)
    {

        u_cap_v = ultra_cap_v_sense();
        hdg_rect_v = hdg_rect_v_sense();

        //para C#
        u_cap_v_send = (unsigned int)(u_cap_v*1000);
        hdg_rect_v_send = (unsigned int)(hdg_rect_v*1000);

        //enviar tensao do cap
        if(u_cap_v_send>=1000) printf("%d",u_cap_v_send);
        else if(u_cap_v_send>=100) printf("0%d",u_cap_v_send);
        else if(u_cap_v_send>=10) printf("00%d",u_cap_v_send);
        else printf("000%d",u_cap_v_send);
        //enviar tensao do hdg
        if(hdg_rect_v_send>=1000) printf("%d\n",hdg_rect_v_send);
        else if(hdg_rect_v_send>=100) printf("0%d\n",hdg_rect_v_send);
        else if(hdg_rect_v_send>=10) printf("00%d\n",hdg_rect_v_send);
        else printf("000%d\n",hdg_rect_v_send);
    }
}
```

C# code

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.IO.Ports;

public class interf_cap : Form
{
    private Button btnport;
    private PictureBox pictureBox1;
    private System.IO.Ports.SerialPort serialPort1;
    private System.ComponentModel.IContainer components;
    private TextBox txtvolt;
    private TextBox txtenergy;
    private TextBox txtpercent;
    private Label label1;
    private Label label2;
    private TextBox txtcap_read;
    private TextBox txtvolt_read;
    private Button btnExit;
    private Timer timer1;
    private TextBox txtcurrent;
    private ProgressBar pgrcap;
    #region Windows code

    private void InitializeComponent()
    {
    #endregion

    Boolean flag;
    double capacity, max_volt, max_energy, cap_volt, cap_energy;
    double percent, hdg_volt, current;
    string in_buff;

    public interf_cap()
    {
        InitializeComponent();
    }
    public static void Main()
    {
        interf_cap main = new interf_cap();
        Application.Run(main);
    }

    private void pictureBox1_Click(object sender, EventArgs e)
    {

    }
```

```
private void button1_Click(object sender, EventArgs e)
{
    flag = double.TryParse(txtcap_read.Text, out capacity);
    if (flag == false)
    {
        MessageBox.Show("Only numbers", "Input Error");
        return;
    }

    flag = double.TryParse(txtvolt_read.Text, out max_volt);
    if (flag == false)
    {
        MessageBox.Show("Only numbers", "Input Error");
        return;
    }
    max_energy = 0.5 * capacity * max_volt * max_volt;
    serialPort1.Open();

    timer1.Start();
}

private void btnExit_Click(object sender, EventArgs e)
{
    Close();
}

private void timer1_Tick_1(object sender, EventArgs e)
{
    in_buff = null;

    cap_volt = 0;

    do
    {
        in_buff = serialPort1.ReadLine();
    } while (in_buff.Length != 8);

    serialPort1.DiscardInBuffer();
```

```
            cap_volt = Convert.ToDouble(in_buff.Substring(0,4)) / 1000;
            hdg_volt = Convert.ToDouble(in_buff.Substring(4,4)) / 1000;
            current = Math.Abs(hdg_volt - cap_volt);
            cap_energy = 0.5 * capacity * cap_volt * cap_volt;
            txtcurrent.Text = "Charge Current: " + current.ToString() + " A";
            txtvolt.Text = "Ultra Cap Voltage: " + cap_volt.ToString() + " V";
            txtenergy.Text = "Ultra Cap Energy: " + cap_energy.ToString() + " J";

            percent = cap_energy * 100 / max_energy;

            txtpercent.Text = "Charge percentage: "+ percent.ToString() + "%";

            pgrcap.Value = (int)percent;
        }

    }
```

Software – microcontroller water meter

myGSM.h

```
#define BUFFER 100
#define SMS_LONG 30
#define ctrl_z 0x1A

unsigned int i;
unsigned char rx_buffer[BUFFER];
unsigned char buffer_temp[BUFFER];
unsigned char prompt_sms[5]={13,10,'>',32};
unsigned char ok[7]={13,10,'O','K',13,10};
unsigned char rx_index=0, temp, inicio, envio;
unsigned char sms[SMS_LONG];
unsigned char num_dest[]="\"+351963544085\"";


void putch(unsigned char byte)
{
    /* output one byte */
    while(!PIR1bits.TXIF)
        continue;
    TXREG = byte;
}

void limpa_buffer(void)
{
    rx_index=0;
    if (RCSTAbits.OERR)
    {
        RCSTAbits.CREN=0;
        RCSTAbits.CREN=1;
        temp=RCREG;
        temp=RCREG;
    }
    if (RCSTAbits.FERR) temp=RCREG;
    for (i=0;i<BUFFER;i++) rx_buffer[i]='\0';
}

void limpa_sms(void) {for (i=0;i<SMS_LONG;i++) sms[i]='\0'; }


void limpa_buffer_temp(void)
{
    for (i=0;i<BUFFER;i++) buffer_temp[i]='\0';
}
```

```
void GsmOn(void)
{
    PORTBbits.RB3=1;
    for (i=0;i<4;i++) Delay10KTCYx(50); //aprox 250Ms
    PORTBbits.RB3=0;
}

void envio_sms(const char *texto)
{
    envio=0;
    limpa_buffer();
    i=0;
    do{
        limpa_buffer();
        //printf("at+cmgs=\"+351963544085\"\r");
        printf("at+cmgs=%s\r",num_dest);
        Delay10KTCYx(50);
        i++;
    }while((memcmp(prompt_sms,rx_buffer,4)!=0)&&i<50);

    if (memcmp(prompt_sms,rx_buffer,4)==0)
    {
        limpa_buffer();
        printf("%s",texto);
        putch(ctrl_z);

        for (i=0;i<100;i++) // 25s p/ timeout
        {
            Delay10KTCYx(50);
            if (rx_buffer[4]!='\0') break;
        }

        Delay10KTCYx(50);
        if (memcmp(ok,&rx_buffer[strlen(rx_buffer)-6],6)==0) flagSms=1;

    }
    limpa_buffer();
    printf("at+cmgd=1,4\r");
    Delay10KTCYx(50);
    limpa_buffer();
}


void init_gsm(void)
{
//**************iniciar modulo**********************************
//******desligar eco e testar comunicação***********************
//******activar sms em modo texto*******************************
//******por parâmetros de sms at+csdh=0*************************
```

```
inicio=0;
    while (inicio==0)
    {
            GsmOn();
            printf("ate\r\n"); // desliga eco
            Delay10KTCYx(50);
            limpa_buffer();
            printf("at\r\n"); // testa comunicação
            Delay10KTCYx(50);
            if (memcmp(ok,rx_buffer,6)==0)
            {
                for (i=0;i<25;i++) Delay10KTCYx(100);
                //tempo para inicialização do sim card
                limpa_buffer();
                printf("at+csdh=0\r"); // recepção de sms
                Delay10KTCYx(50);
                if (memcmp(ok,rx_buffer,6)==0)
                {
                    limpa_buffer();
                    printf("at+cmgf=1\r"); // sms em modo texto
                    Delay10KTCYx(50);
                    if (memcmp(ok,rx_buffer,6)==0)
                    {
                        limpa_buffer();
                        printf("at+cnmi=0,0,0,0,0\r"); //0,0,0,0,0
                        Delay10KTCYx(50);
                        if (memcmp(ok,rx_buffer,6)==0)
                        {
                            inicio=1;
                        }

                    }
                }
            }
        }

}
```

eeprom.h

```
void eepromWrite(unsigned char addr,unsigned char eeData)
{
    EEADR=addr;
    EEDATA=eeData;
    EECON1bits.EEPGD=0;
    EECON1bits.CFGS=0;
    EECON1bits.WREN=1;
    INTCONbits.GIEL=0;
    INTCONbits.GIEH=0;
    EECON2=0x55;
    EECON2=0x0AA;
    EECON1bits.WR=1;
    INTCONbits.GIEL=1;
    INTCONbits.GIEH=1;
    while(EECON1bits.WR);
    EECON1bits.WREN=0;
}

unsigned char eepromRead(unsigned char addr)
{
    unsigned char data;

    EEADR=addr;
    EECON1bits.EEPGD=0;
    EECON1bits.CFGS=0;
    EECON1bits.RD=1;
    data=EEDATA;

    return data;
}
```

display_cmd_nibble_hardware.h

```
#define DB4 PORTCbits.RC0
#define DB5 PORTCbits.RC1
#define DB6 PORTCbits.RC2
#define DB7 PORTCbits.RC3

#define RS PORTCbits.RC5
#define RW PORTCbits.RC4
#define E  PORTBbits.RB0
```

display_cmd_nibble.h

```
#include <string.h>
#include <stdio.h>
#include "display_cmd_nibble_hardware.h"

#define LINE_LIMIT 16

unsigned char lcdString[25];
unsigned int i=0;

/*******************************************************/
void busyDelay(void)
{
    Delay1KTCYx(3);
    Delay100TCYx(2);
}
/*******************************************************/
void instDelay(void)
{
    Delay100TCYx(1); //1 // 40us
}
/*******************************************************/
void clearLcdString(void)
{
    for (i=0;i<25;i++) lcdString[i]='\0';
}

/*******************************************************/
void lcdStrobe(void)
{
    //E=1; DelayUs(1); E=0; DelayUs(1);
    E=1;
    Delay1TCY(); Delay1TCY(); Delay1TCY();
    E=0;
    Delay1TCY(); Delay1TCY(); Delay1TCY();
}
```

```
/*****************************************************/
void checkBusy(void)
{
    unsigned char BF=0;

    TRISC=TRISC | 0x0F;   /*1=IN 0=OUT*/
    RS=0; RW=1;
    lcdStrobe();
    if (DB7) instDelay();
    BF=DB7;
        lcdStrobe();
    RW=0;
    TRISC=TRISC & 0xF0;  /*1=IN 0=OUT*/
}

//*****************************************************
void clearDisplay(void)
{
    RS=0; RW=0;
    PORTC=PORTC & 0xF0;
    lcdStrobe();
    //busyDelay();
    PORTC=PORTC | 0x01;
    lcdStrobe();
    busyDelay();
    ClrWdt();
 }
/*************************************************/
void returnHome(void)
{
    RS=0; RW=0;
    PORTC=PORTC & 0xF0;
    lcdStrobe();
    //busyDelay();
    PORTC=PORTC | 0x02;
    lcdStrobe();
    busyDelay();
    ClrWdt();
 }
```

```
/*****************************************************/
void setModeIncrementShift(unsigned char inc, unsigned char shift)
{
    inc=inc<<1;
    RS=0; RW=0;
    PORTC=PORTC & 0xF0;
    lcdStrobe();
    //busyDelay();
    PORTC=((PORTC| 0x04) | inc) | shift;
    lcdStrobe();
    //busyDelay();
    ClrWdt();
 }
/*****************************************************/
//Display ON/OFF, cursor, blink
void setOnCursorBlink(unsigned char on, unsigned char cursor, unsigned char blink)
{
    on=on<<2;
    cursor=cursor<<1;
    RS=0; RW=0;
    PORTC=PORTC & 0xF0;
    lcdStrobe();
    //busyDelay();
    PORTC=(((PORTC | 0x08) | on) | cursor) | blink;
    lcdStrobe();
    ClrWdt();
}
/*********************************************************************/
void setInterfaceLineFont(unsigned char set) //Function set
{
    RS=0; RW=0;
    PORTC=(PORTC & 0xF0) | (set>>4);
    lcdStrobe();
    //busyDelay();
    PORTC=(PORTC & 0xF0) | (set & 0x0F);
    lcdStrobe();
    ClrWdt();
}
```

```
void writeDataRAM(unsigned char digit)
{
    RS=1; RW=0;
    PORTC=(PORTC & 0xF0) | (digit>>4);
    lcdStrobe();
    PORTC=(PORTC & 0xF0) | (digit & 0x0F);
    lcdStrobe();
    RS=0;
    checkBusy();
    ClrWdt();
}
/*****************************************************************************/
void setDDRAM_AddressLineDigit(unsigned char position)
{
    RS=0; RW=0;
    PORTC=((PORTC & 0xF0) | 0x08) | (position>>4);
    lcdStrobe();
    PORTC=(PORTC & 0xF0) | (position & 0x0F);
    lcdStrobe();
    checkBusy();
    ClrWdt();
}
/*****************************************************************************/
void setCGRAM_Address(unsigned char CG)
{
    RS=0; RW=0;
    PORTC=((PORTC & 0xF0) | 0x04) | (CG>>4);
    lcdStrobe();
    PORTC=(PORTC & 0xF0) | (CG & 0x0F);
    lcdStrobe();
    checkBusy();
    ClrWdt();
}
/*****************************************************************************/
void displayInit(void)
{
    Delay1KTCYx(150);
    //ClrWdt();
    //----------------------------------------
    RS=0; RW=0;
```

```
    PORTC=(PORTC & 0xF0) | 0x03;
    lcdStrobe();
    ClrWdt();
    //---------------------------------------
    Delay1KTCYx(25);
    setInterfaceLineFont(0x28);
    Delay1KTCYx(25);
    setInterfaceLineFont(0x28);
    Delay1KTCYx(25);
    setOnCursorBlink(1,0,0);
    Delay1KTCYx(25);
    clearDisplay();
    Delay1KTCYx(25);
    setModeIncrementShift(1,0);
    Delay1KTCYx(25);
    //ClrWdt();
}
/*********************************************************************/
void lcdPrint(unsigned char line, unsigned char col, const char *data)
{
    unsigned char lineIndex;

    //----------Se display de 4 linhas-------------------
    //if (line==1) setDDRAM_AddressLineDigit(0+col);
    //else if (line==2) setDDRAM_AddressLineDigit(64+col);
    //else if (line==3) setDDRAM_AddressLineDigit(20+col);
    //else if (line==4) setDDRAM_AddressLineDigit(84+col);

    //----------Se display de 2 linhas-------------------
    setDDRAM_AddressLineDigit((line-1)*64 + col);

    for (lineIndex=0;lineIndex<strlen(data);lineIndex++)
    {
        writeDataRAM(data[lineIndex]);
        //busyDelay();
    }
    while(lineIndex<LINE_LIMIT)
    {
        writeDataRAM(0x20); //0x20 = SPACE
```

```
        //busyDelay();
        lineIndex++;
    }
}
/*************************************************************************/
void lcdPrintChar(unsigned char line, unsigned char col, char dataChar)
{
    //----------Se display de 4 linhas-------------------
    //if (line==1) setDDRAM_AddressLineDigit(0+col);
    //else if (line==2) setDDRAM_AddressLineDigit(64+col);
    //else if (line==3) setDDRAM_AddressLineDigit(20+col);
    //else if (line==4) setDDRAM_AddressLineDigit(84+col);

    //---------Se display de 2 linhas--------------------
    setDDRAM_AddressLineDigit((line-1)*64 + col -8); //
    //busyDelay();

    writeDataRAM(dataChar);
    //busyDelay();
}
/*************************************************************************/
void initGraph(void)
{
    setCGRAM_Address(0x00);
    writeDataRAM(0x00);
    writeDataRAM(0x1D);
    writeDataRAM(0x0F);
    writeDataRAM(0x1F);
    writeDataRAM(0x1E);
    writeDataRAM(0x17);
    writeDataRAM(0x00);
    writeDataRAM(0x00);
}
/*************************************************************************/
unsigned char displayOff(void)
{
    setOnCursorBlink(0,0,0);
    return 0;
}
```

```
/*************************************************************************/
unsigned char displayOn(void)
{
    setOnCursorBlink(1,0,0);
    return 1;
}
/*************************************************************************/
void testDisplay(void)
{
    clearDisplay();
    sprintf(lcdString,"SOLZAIMA"),
    lcdPrint(1,0,lcdString);
    clearLcdString();
}
```

water_meter_1_8.c

```
#include <p18f2520.h>
#include <stdio.h>
#include <delays.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <usart.h>
#include "eeprom.h"
#include "display_cmd_nibble.h"
#include "myGSM.h"

//-----------------------------------------------
//-----------------Pré processador-------------
//-----------------------------------------------

//-----------------------------------------------
//---------Definiçao dos canais da ADC-----------
//-----------------------------------------------


//-----------------------------------------------
//-------------Variables-------------------------
//-----------------------------------------------
int tick50Ms=0;
int countSeg=0;
int freqSeg=0;
int flagTmr1=0;
int countTmr3=0;
char smsTimeOut=0;
unsigned int freqAcum=0;
float freqAcumMed=0, litrosMin=0, litrosTotal=0, caudal, consumoTotal, consumoTotalLcd;
unsigned char *ptr = (unsigned char*)&consumoTotal;
unsigned char *ptrLcd = (unsigned char*)&consumoTotalLcd;
float ultraCapVdd;
char* frase;
char flagSms=0;
char flagStop=1;
//-----------------------------------------------
//-------------Functions-------------------------
//-----------------------------------------------
void Init(void);
void InterruptHandlerHigh (void);
void InterruptHandlerLow (void);
void enable5V(void);
void disable5V(void);
void enable3V8(void);
void disable3V8(void);
float readUltraCapVdd(void);
void goSleep(void);
void enableRS232Rx(void);
void disableRS232Rx(void);
void enableRS232Tx(void);
void disableRS232Tx(void);
void enableLevelShifter(void);
void enableIntrs(void);
void disableIntrs(void);
```

```
void Init(void)
{
    INTCONbits.GIE = 0;     //disables general interrupts
    INTCONbits.GIEH = 0;    //disables high interrupts
    INTCONbits.GIEL = 0;    //disables low interrupts

    //OSCILLATOR CONFIG
    OSCCONbits.IDLEN=0;//enter sleep mode on SLEEP instruction
    OSCCONbits.SCS0=0; //primary oscillator
    OSCCONbits.SCS1=0; //primary oscillator

    //--------NO CRYSTAL MODE-------
    OSCCONbits.IRCF2=1;//internal 8Mhz Osc
    OSCCONbits.IRCF1=1;
    OSCCONbits.IRCF0=1;
    //--------------------------------

    //PORT CONFIG
    TRISA=0b00001001;
    //RA0 analog intput - PIC_VDD_READ
    //RA1 digital output - Vref_ACTIVE
    //RA2 digital output - ADC_PNP_READ
    //RA3 analog intput - Vref+
    //RA4 digital output - 3V8_REG_ENABLE
    //RA5 digital output - 5V_REG_ENABLE
    //RA6 & RA7 CRISTAL
    TRISB=0b00000110;
    //RB0 digital output - LCD_ENABLE
    //RB1 digital input - HDG_INT
    //RB2 digital input - LCD_INT
    //RB3 digital output - POWER_GSM
    //RB6 & RB7 - PGC & PGD
    TRISC=0b10000000;
    //RC0 digital input/output - D4
    //RC1 digital input/output - D5
    //RC2 digital input/output - D6
    //RC3 digital input/output - D7
    //RC4 digital output - RW
    //RC5 digital output - RS
    //RC6 digital output - PIC_GSM_TX
    //RC7 digital input - PIC_GSM_RX

    // DIGITAL/ANALOG PORTS CONFIG
    ADCON1bits.PCFG3=1;
    ADCON1bits.PCFG2=1;
    ADCON1bits.PCFG1=1;
    ADCON1bits.PCFG0=0; //AN0 is ANALOG, all other ports are DIGITAL

    //ADC CONFIG
    ADCON0bits.ADON=0; //ADC disabled
    ADCON0bits.GO=0;  //ADC off
    ADCON1bits.VCFG1=0; //Negative reference Vss
    ADCON1bits.VCFG0=1; //Positive reference Vref+ - LMV431A

    //SHUTDOWNS
    PORTAbits.RA1=0;    //shutdown LMV431
    PORTAbits.RA2=1;    //shutdown PNP - base voltage = Vdd
    PORTAbits.RA4=0;    //shutdown 3V8 Reg
    PORTAbits.RA5=0;    //shutdown 5V Reg
    PORTBbits.RB0=0;    //LCD_ENABLE low
    PORTBbits.RB3=0;    //POWER_GSM low
    PORTBbits.RB4=0;    //Level Shifter VccB low
    PORTCbits.RC0=0;    //D4 low
    PORTCbits.RC1=0;    //D5 low
    PORTCbits.RC2=0;    //D6 low
    PORTCbits.RC3=0;    //D7 low
    PORTCbits.RC4=0;    //RW low
    PORTCbits.RC5=0;    //RS low
    PORTCbits.RC6=0;    //PIC_GSM_TX low
```

```
//INTERRUPT CONFIG
INTCONbits.PEIE = 1;    //peripheral interrupts enabled
RCONbits.IPEN = 1;      //enable priority levels
INTCON2bits.RBPU=1;     //Portb pull ups disabled
INTCONbits.RBIE=0;      //port B on change disabled
    //INT1 - HDG_INT
INTCON2bits.INTEDG1 = 0;//INT1 (HDG_INT) - interrupt on falling edge
INTCON3bits.INT1IE = 1; //INT1 external interrupt enable - HDG_INT interrupt
INTCON3bits.INT1IP = 0; //INT1 low priority - HDG_INT interrupt
INTCON3bits.INT1IF=0;   //clear interrupt flag
    //INT2 - LCD_INT
INTCON2bits.INTEDG2=1;  //INT2 (LCD_INT) - interrupt on rising edge
INTCON3bits.INT2IP=0;   //INT2 low priority - LCD_INT interrupt
INTCON3bits.INT2IE=1;   //INT2 external interrupt enable - LCD_INT interrupt
INTCON3bits.INT2IF=0;   //clear interrupt flag
    //TIMER 0
INTCONbits.TMR0IE = 1;  //TMR0 overflows interrupt enable
INTCON2bits.TMR0IP = 1; //TMR0 high priority interrupt
INTCONbits.TMR0IF = 0;  //TMR0 clear interrupt flag
    //TIMER 1
PIE1bits.TMR1IE = 1;    //TMR1 overflow interrupt enable
IPR1bits.TMR1IP = 1;    //TMR1 high priority interrupt
PIR1bits.TMR1IF = 0;    //TMR1 clear interrupt flag
    //TIMER 2
PIE1bits.TMR2IE = 1;    //TMR2 overflow interrupt enable
IPR1bits.TMR2IP = 1;    //TMR2 high priority interrupt
PIR1bits.TMR2IF = 0;    //TMR2 clear interrupt flag
    //TIMER 3
PIE2bits.TMR3IE=0;      //TMR3 overflow interrupt disabled
IPR2bits.TMR3IP=0;      //TMR 3 low priority interrupt
PIR2bits.TMR3IF=0;      //TMR3 clear interrupt flag
    //RX
PIE1bits.RCIE=0;        //UART RX interrupt disabled
PIR1bits.RCIF=0;        //clean interrupt flag
IPR1bits.RCIP=1;        //high priority
//USART CONFIG
TXSTAbits.TX9 = 0;      //8 bit transmission
TXSTAbits.TXEN = 0;     //transmitter disabled initially
TXSTAbits.SYNC = 0;     //asynchronous mode
TXSTAbits.BRGH = 1;     //high speed transmission
RCSTAbits.SPEN = 0;     //serial port disabled initially
BAUDCONbits.BRG16 = 1;  //16 bit baud rate generator
SPBRG=8;                //115200 baud rate - compatible with GSM module baud rate
RCSTAbits.CREN=0;       //disables receiver

//TIMER0 CONFIG
T0CONbits.T08BIT = 1; //8 bit counter
T0CONbits.T0CS = 0; //TMR0 internal instruction clock -> 4Mhz
T0CONbits.T0SE = 1; //high to low transition
T0CONbits.PSA = 0; //prescaler active
T0CONbits.T0PS2 = 1;//prescaler 256 -> TMR0L overflows after 65ms
T0CONbits.T0PS1 = 1;//prescaler 256 -> TMR0L overflows after 65ms
T0CONbits.T0PS0 = 1;//prescaler 256 -> TMR0L overflows after 65ms
INTCONbits.TMR0IF = 0; //clear interrupt flag
T0CONbits.TMR0ON=0; //tmr0 off

//TIMER1 CONFIG
T1CONbits.RD16=1;   //16 bit operation
T1CONbits.T1CKPS0=1;
T1CONbits.T1CKPS1=1; //1:8 prescaler
T1CONbits.TMR1CS=0;  //fosc/4
```

```
    //TIMER2 CONFIG
    T2CONbits.T2OUTPS0=1;
    T2CONbits.T2OUTPS1=1;
    T2CONbits.T2OUTPS2=1;
    T2CONbits.T2OUTPS3=1;
    T2CONbits.T2CKPS0=1;
    T2CONbits.T2CKPS1=1;
    // prescale=16 postscale=16
    // Fosc=4MHz Fcy=1Mhz Tcy=1us each_interrupt=prescale*postscale*1us*PR2=50ms
    PR2=195; //50ms
    PIR1bits.TMR2IF=0;      //clear interrupt flag
    T2CONbits.TMR2ON=0;     //tmr2 off


    //TIMER3 CONFIG
    T3CONbits.RD16=1;       //16 bit operation
    T3CONbits.T3CKPS1=1;
    T3CONbits.T3CKPS0=1;    //tmr3 1:8 prescaler
    T3CONbits.TMR3CS=0;     //fosc/4
    T3CONbits.TMR3ON=0;     //tmr 3 disabled

    INTCONbits.GIEH = 1;    //enables high interrupts
    INTCONbits.GIEL = 1;    //enables low interrupts
    INTCONbits.GIE = 1;     //enables general interrupts
}

void enableRS232Tx(void)
{
    TXSTAbits.TXEN = 1; //enable serial transmitter
    RCSTAbits.SPEN = 1; //enable serial port
}

void disableRS232Tx(void)
{
    TXSTAbits.TXEN = 0; //disable  transmitter
    RCSTAbits.SPEN = 0; //disable serial port
    PORTCbits.RC6=0;    //RC6 low (PIC_GSM_TX)
}
void enableRS232Rx(void)
{
    RCSTAbits.CREN=1;   //enables receiver
    PIE1bits.RCIE=1;    //enable interrupt
}

void disableRS232Rx(void)
{
    RCSTAbits.CREN=0;   //disable receiver
    PIE1bits.RCIE=0;    //disable interrupt
}

void enable5V(void)
{
    PORTAbits.RA5=1;  //5V_REG_ENABLE high
    Delay10KTCYx(1);  //stabilization delay
}

void disable5V(void)
{
    PORTAbits.RA5=0; //5V_REG_ENABLE low
}

void enable3V8(void)
{
    PORTAbits.RA4=1; //3V8_REG_ENABLE high
    Delay10KTCYx(1); //stabilization delay
}
```

```
void enableLevelShifter(void)
{
    PORTBbits.RB4=1;
}

void disableLevelShifter(void)
{
    PORTBbits.RB4=0;
}

void enableIntrs(void)
{
    INTCON3bits.INT1IE=1;   //INT1 external interrupt enable - HDG_INT interrupt
    INTCON3bits.INT2IE=1;   //INT2 external interrupt enable - LCD_INT interrupt
    INTCONbits.TMR0IE = 1;  //TMR0 overflows interrupt enable
    PIE1bits.TMR1IE = 1;    //TMR1 overflow interrupt enable
    PIE1bits.TMR2IE = 1;    //TMR2 overflow interrupt enable
    PIE2bits.TMR3IE=0;      //TMR3 overflow interrupt disable
}

void disableIntrs(void)
{
    INTCON3bits.INT1IE=0;   //INT1 external interrupt disable - HDG_INT interrupt
    INTCON3bits.INT2IE=0;   //INT2 external interrupt disable - LCD_INT interrupt
    INTCONbits.TMR0IE = 0;  //TMR0 overflows interrupt disable
    PIE1bits.TMR1IE = 0;    //TMR1 overflow interrupt disable
    PIE1bits.TMR2IE = 0;    //TMR2 overflow interrupt disable
    PIE2bits.TMR3IE=1;      //TMR3 overflow interrupt enable
}

float readUltraCapVdd(void)
{
    float result[5], resultMed=0;
    int i;

    PORTAbits.RA1=1;    //LMV431 high (active for external Vref+)
    PORTAbits.RA2=0;    //PNP low (active for UltraCap voltage reading)
    //ADC CONFIG
    ADCON2=0b00111101;      //Left justified  20Tad  Fosc/16
    ADCON0bits.CHS3=0;
    ADCON0bits.CHS2=0;
    ADCON0bits.CHS1=0;
    ADCON0bits.CHS0=0;      //AN1
    ADCON0bits.ADON=1;      //ADC enabled

    for(i=0;i<5;i++)
    {
        ADCON0bits.GO=1;    //start conversion
        while(ADCON0bits.GO);
        result[i] = (float)(((float)(ADRESH)*1.24)/256);
        result[i]=result[i]*12.2/2.2+0.1;//0.1 is the colector emitter drop
    }//5 readings cycle

    ADCON0bits.ADON=0;  //turn off ADC
    PORTAbits.RA1=0;    //shutdown LMV431
    PORTAbits.RA2=1;    //shutdown PNP

    for(i=0;i<5;i++)
    {
        resultMed=resultMed+result[i];
    }//calculate average result

    return (resultMed/5);
}
```

```
void goSleep(void)
{
    _asm    SLEEP
    _endasm
}

//-----------------------------------------------------------
//-----------Low priority interrupt vector-----------------
//-----------------------------------------------------------
#pragma code InterruptVectorLow = 0x18
void InterruptVectorLow (void)
{
    _asm
    goto InterruptHandlerLow //jump to interrupt routine
    _endasm
}
#pragma code

#pragma interruptlow InterruptHandlerLow
void InterruptHandlerLow ()
{
        if(!INTCON3bits.INT1IF&&!INTCON3bits.INT2IF&&!PIR2bits.TMR3IF) goSleep();

        if(PIR2bits.TMR3IF)//30 sec sms timeout
        {
            countTmr3++;

            if(countTmr3>57)
            {
                T3CONbits.TMR3ON=0;     //tmr 3 disabled
                smsTimeOut=1;
                countTmr3=0;
                disableIntrs();
            }

            PIR2bits.TMR3IF=0;
        }

        if(INTCON3bits.INT1IF)//HDG_INT interrupt
        {
            if((!T2CONbits.TMR2ON)&&(!T0CONbits.TMR0ON))
            {
                flagStop=0;
                litrosMin=0;
                litrosTotal=0;
                freqSeg=0;
                freqAcum=0;
                freqAcumMed=0;
                countSeg=0;
                tick50Ms=0;
                TMR0L=0;//clear TMR0
                T2CONbits.TMR2ON=1;
                T0CONbits.TMR0ON=1;
            }

            freqSeg++; //update counter
            INTCON3bits.INT1IF=0; //clear interrupt flag
            TMR0L=0;//reset timer
        }
```

```
        if(INTCON3bits.INT2IF)//LCD_INT interrupt
        {
                INTCON3bits.INT2IF=0; //clear interrupt flag

                clearLcdString();
                *(ptrLcd+3) = eepromRead(0x00); //read values from EEPROM
                *(ptrLcd+2) = eepromRead(0x01);
                *(ptrLcd+1) = eepromRead(0x02);
                *ptrLcd = eepromRead(0x03);

                if(consumoTotalLcd<10)//format string
                {
                    lcdString[0]='0'+(int)consumoTotalLcd;
                    lcdString[1]='.';
                    lcdString[2]='0'+((int)(consumoTotalLcd*10))%10;
                    lcdString[3]='0'+((int)(consumoTotalLcd*100))%10;
                } else if(consumoTotalLcd<100)
                    {

                        lcdString[0]='0'+(int)consumoTotalLcd/10;
                        lcdString[1]='0'+(int)consumoTotalLcd%10;
                        lcdString[2]='.';
                        lcdString[3]='0'+((int)(consumoTotalLcd*10))%10;
                        lcdString[4]='0'+((int)(consumoTotalLcd*100))%10;
                    } else
                        {
                            lcdString[0]='0'+(int)consumoTotalLcd/100;
                            lcdString[1]='0'+(int)((int)consumoTotalLcd%100)/10;
                            lcdString[2]='0'+(int)((int)consumoTotalLcd%100)%10;
                            lcdString[3]='.';
                            lcdString[4]='0'+((int)(consumoTotalLcd*10))%10;
                            lcdString[5]='0'+((int)(consumoTotalLcd*100))%10;
                        }

                enable5V();
                displayInit();
                clearDisplay();
                lcdPrint(1,0,lcdString);
                clearLcdString();
                T1CONbits.TMR1ON=1; //start TMR1 for LCD shutdown
        }
}
//-------------------------------------------------------------
//-----------High priority interrupt vector----------------
//-------------------------------------------------------------
#pragma code InterruptVectorHigh = 0x08 // Compatibility mode
void InterruptVectorHigh (void)
{
    _asm
    goto InterruptHandlerHigh //jump to interrupt routine
    _endasm
}
#pragma code

#pragma interrupt InterruptHandlerHigh
void InterruptHandlerHigh ()
{
        if(!INTCONbits.TMR0IF&&!PIR1bits.TMR1IF&&!PIR1bits.TMR2IF&&!PIR1bits.RCIF) goSleep();
```

```
if(PIR1bits.RCIF)
{
    rx_buffer[rx_index]=getch();
    rx_index=rx_index+1;
    if (rx_index>(BUFFER-2)) rx_index=(BUFFER-2);
    PIR1bits.RCIF=0;
}

if(INTCONbits.TMR0IF)//water flow off detection - TMR0 overflow
{
    T2CONbits.TMR2ON=0; //shutdown TMR2
    T0CONbits.TMR0ON=0; //shutdown TMR0
    INTCONbits.TMR0IF=0; //clear interrupt flag
    PIR1bits.TMR2IF=0;   //clear interrupt flag
    flagStop=1;
    if(countSeg) //if more than one second passed, do the math
    {
        freqAcumMed=(float)freqAcum/(float)countSeg;
        litrosMin=((freqAcumMed+160)/440)*(float)countSeg/60;
        litrosTotal=litrosTotal+litrosMin;
        ultraCapVdd=readUltraCapVdd();
        enableRS232Tx();

        *(ptr+3) = eepromRead(0x00);
        *(ptr+2) = eepromRead(0x01);
        *(ptr+1) = eepromRead(0x02);
        *ptr = eepromRead(0x03);
        consumoTotal=consumoTotal+litrosTotal;
        eepromWrite(0x00,*(ptr+3));
        eepromWrite(0x01,*(ptr+2));
        eepromWrite(0x02,*(ptr+1));
        eepromWrite(0x03,*ptr);

        disableRS232Tx();
    }
}

if(PIR1bits.TMR1IF)//LCD shutdown - TMR1 overflow
{
    if(flagTmr1==10) //aproximatelly 5 seconds have passed
    {
        T1CONbits.TMR1ON=0; //TMR1 off
        E=0;   //LCD_ENABLE low
        DB4=0; //all data ports low
        DB5=0;
        DB6=0;
        DB7=0;
        RS=0;  //RS low
        RW=0;  //RW low
        disable5V();
        flagTmr1=0;  //clear flag
        PIR1bits.TMR1IF=0; //clear TMR1 interrupt flag
        goSleep();
    } else
        {
            flagTmr1++; //increase Tmr1flag
            PIR1bits.TMR1IF=0; //clear TMR1 interrupt flag
        }
}
```

```
        if(PIR1bits.TMR2IF)
        {
            tick50Ms++;
            if(tick50Ms==20) //1 second passed
            {
                countSeg++;
                caudal=(float)(freqSeg+160)/440;
                enableRS232Tx();

                freqAcum=freqAcum+freqSeg;
                disableRS232Tx();
                freqSeg=0;
                tick50Ms=0;
                TMR0L=0;

                if(countSeg==60)//1 minute passed
                {
                    freqAcumMed=(float)(freqAcum/countSeg);
                    litrosMin=(freqAcumMed+160)/440;
                    litrosTotal=litrosTotal+litrosMin;
                    enableRS232Tx();

                    disableRS232Tx();
                    freqAcum=0;
                    litrosMin=0;
                    countSeg=0;
                }
            }
            PIR1bits.TMR2IF=0;
        }
}

void main (void)
{
    Init();

    for(;;)
    {
        if((ultraCapVdd>4.4)&&!flagSms&&flagStop)
        {

            disableIntrs();
            enableLevelShifter();
            enable3V8();
            enableRS232Tx();
            enableRS232Rx();
            T3CONbits.TMR3ON=1;     //tmr 3 enabled
            GsmOn();
            init_gsm();
            sprintf(sms,"ah e tal testezinho");
            envio_sms(sms);
            disableRS232Rx();
            disableRS232Tx();
            disableLevelShifter();
            disable3V8();
            T3CONbits.TMR3ON=0;     //tmr 3 disabled
            flagSms=1;
        }
```

```
        if(smsTimeOut&&flagStop)
        {
            flagSms=0;
        }

        if(!flagSms&&flagStop)
        {
            goSleep();
            Nop();
        }
    }
}
```